# Learning sequential force interaction skills

Manschitz, Simon; Gienger, Michael; Kober, Jens; Peters, Jan

**DOI**
10.3390/ROBOTICS9020045

**Publication date**
2020

**Document Version**
Final published version

**Published in**
Robotics

*robotics*

MDPI

*Article*

# Learning Sequential Force Interaction Skills

**Simon Manschitz [1,*], Michael Gienger [1], Jens Kober [2] and Jan Peters [3,4]**

1 Honda Research Institute Europe, 63073 Offenbach, Germany; michael.gienger@honda-ri.de
2 Cognitive Robotics Department, Delft University of Technology, 2628 CD Delft, The Netherlands; j.kober@tudelft.nl
3 Institute for Intelligent Autonomous Systems, Technische Universität Darmstadt, 64289 Darmstadt, Germany; mail@jan-peters.net
4 Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany
* Correspondence: simon.manschitz@honda-ri.de

Received: 2 April 2020; Accepted: 15 June 2020; Published: 17 June 2020


**Abstract:** Learning skills from kinesthetic demonstrations is a promising way of minimizing the gap between human manipulation abilities and those of robots. We propose an approach to learn sequential force interaction skills from such demonstrations. The demonstrations are decomposed into a set of movement primitives by inferring the underlying sequential structure of the task. The decomposition is based on a novel probability distribution which we call Directional Normal Distribution. The distribution allows infering the movement primitive's composition, i.e., its coordinate frames, control variables and target coordinates from the demonstrations. In addition, it permits determining an appropriate number of movement primitives for a task via model selection. After finding the task's composition, the system learns to sequence the resulting movement primitives in order to be able to reproduce the task on a real robot. We evaluate the approach on three different tasks, unscrewing a light bulb, box stacking and box flipping. All tasks are kinesthetically demonstrated and then reproduced on a Barrett WAM robot.

**Keywords:** human-robot interaction; motor skill learning; learning from demonstration; behavioral cloning

## 1. Introduction

As the variety of industrial tasks is increasing and robots are being more and more deployed in real-world settings, it becomes impossible to pre-program them for all the situations they may encounter. Imitation learning provides powerful methods to reduce the programming effort. By generalizing existing task knowledge for new situations, robots even have the potential to allow the realization of tasks with higher complexity than just by programming alone. Therefore, learning manipulation tasks from human demonstrations has many potential application domains ranging from service robotics to industrial applications.

The contribution of this paper is a concept for learning motor skills for tasks that have the following characteristics: First, the task can be represented by a sequence of object-relative point-to-point movements. Hence, the particular shape and execution speed of a movement are not relevant, but the goal of a movement is. Examples for goals are desired positions or orientations of an end-effector relative to an object in the scene, or the posture of the fingers of a hand. Second, the task involves a physical interaction with the environment. In phases where the robot interacts with an object, it may have to actively apply a force, for instance it may need to push an object. Therefore, movements are not restricted to the kinematic domain. Instead, a goal may also be comprised of a desired force or torque. We refer to tasks that have the aforementioned properties as sequential force interaction tasks.

In this paper, we present a novel motor skill learning framework for point-to-point movements, as well as for their composition. Furthermore, we propose a concept for reactive decision making,

i.e., when to perform which of the movements skills during online execution. Motor skills may comprise force and/or kinematic control magnitudes. Force magnitudes control the interaction force between parts of the robot and the environment, for instance applying a force against a box with the robot's end effector. Kinematic magnitudes control the position/velocity of robot parts, such as for instance the position of the end effector with respect to the above box. This hybrid force/kinematic representation allows learning tasks that are characterized by both kinematic and force interactions. These are of high relevance, for instance in the assembly domains where force control is essential to control the physical interaction.

Our particular focus is learning skills that generalize well to situations which were not demonstrated. Hence, the skill should be independent of absolute object positions, such that it can be generalized to object locations which have not been seen during teaching. Therefore, we demonstrate a task several times with varying object positions. From these demonstrations, the system learns the individual movements and decides for each movement if it should be performed relative to an object and if a force should be applied or not. After learning, we execute the resulting skill on a real robot with a different setup and, therefore, evaluate the generalization capabilities of the skill.

Throughout this paper, we will refer to the individual movements as movement primitives (MPs). The process of extracting attractor MPs representing the single point-to-point movements from the demonstrations will be referred to as task-decomposition. To represent the robot's movements relative to objects in the scene, we use a hybrid position-force task-space controller (see Section 1.2). A task-space is composed of a controlled entity (e.g., end-effector), control variables (e.g., position or force) and a coordinate frame which allows for controlling the robot relative to an object. We pre-define a set of task-spaces which are sufficient for performing a wide range of tasks. Applying our proposed task-decomposition method to the demonstrations then results in a set of MPs, whereas for each MP an appropriate task-space is selected and the most likely attractor goal is found. The basis of the approach is a probability distribution called Directional Normal Distribution. In this paper, we present the distribution and an Expectation-Maximization algorithm for inferring its parameters from data. The task-decomposition was introduced by Manschitz et al. [1]. Compared to this paper, we provide a more in-depth discussion and evaluation of the method.

The task-decomposition results in a set of MPs controlling either the position or orientation of the end-effector or the joint angles of the robot's hand. In a second step, we present a concept for learning to sequence these MPs. When executing a learned skill on a real robot, our system decides at run-time which of the MPs to activate. The system is based on the concept of a sequence graph which was introduced in Manschitz et al. [2,3]. In this paper, we extend the concept to allow for synchronizing the activations of the MPs controlling different entities of the robot (position, orientation, fingers).

In summary, this paper combines and extends two previously published approaches into one consistent sequential skill learning framework. We provide an extended in-depth discussion of the task-decomposition approach, and present additional experiments. Furthermore, we extend the sequence graph concept to allow for synchronizing the activations of the MPs resulting from the task-decomposition. The presented system is capable of learning sequential force interaction skills from only a few demonstrations. We conducted and evaluated experiments to show how the learned skills generalize to configurations of the scene that have never been seen during training.

### 1.1. Related Work

Decomposing tasks based on kinesthetic demonstrations has received a lot of attention in robotics research over recent years. Lioutikov et al. [4] identify the underlying MPs of a task using an Expectation-Maximization (EM) algorithm. The authors assume the demonstrations are pre-segmented and their method prunes out false positive cut points between successive segments. Lemme et al. [5] build a library of MPs from demonstrations. Their approach starts with a single simple MP. By modifying and recombining existing MPs, they are able to represent complex trajectories. Other approaches try to identify MPs using Bayesian Binning [6], Transition State Clustering [7],

Conditional Random Fields [8,9], Gaussian Mixture Models [10–12], Hidden Markov Models [13,14] or Temporal Alignment for Control [15]. Changepoint methods can be used for decomposing a task in a two-step process. After segmenting demonstrations by detecting changepoints [16–21], the individual segments can be assigned to individual MPs. The advantage of such changepoint-based methods is that they can be used for online-analysis of movements. Additionally, the number of MPs is bound by the number of segments and therefore, does not have to be specified in advance.

Recently, also Inverse Reinforcement Learning (IRL) methods have been frequently used in the context of skill identification [22–24]. Instead of directly extracting the MPs from the demonstrations, these methods try to infer reward functions for the (sub)tasks. This kind of approach has two interesting properties. First, instead of decomposing the motions itself, it attempts to capture the reason underlying the motions. Hence, if successful, the resulting models can generalize well to previously unknown situations. Second, the resulting reward functions can be directly used for Reinforcement Learning to improve the model over time afterward. Despite the advantages, one downside of IRL methods is that they are data-intensive, making them impractical for our purpose.

Introducing coordinate frames enables representing motions relative to objects in the scene. Therefore, they can help to improve the generalization capabilities of the system. Niekum et al. [25] segment demonstrations using an auto-regressive Hidden Markov Model. After the segmentation, they extract the coordinate frames of the segments and split the states of the resulting model up to build a Finite State Machine representation of the task. Another approach that explicitly deals with coordinate frames is that of Rozo et al. [26]. The authors use a task-parametrized Gaussian mixture model (TP-GMM) as trajectory representation. A TP-GMM is a hierarchical GMM with two layers. While on the upper layer each mixture represents a MP, the lower layer is used for representing the joint probability of the trajectory (and the velocity) in the different coordinate frames. In addition to learning a task-representation in different coordinate frames, they learn to vary the controller stiffness dynamically. As a result, their approach enables a robot to physically interact with a human co-worker. While human-robot collaboration (e.g., Maeda et al. [27], Caccavale et al. [28], Wu et al. [29]) is an interesting research domain, our approach focuses on tasks where the robot is operating autonomously. A similar approach was proposed by Huang et al. [30]. The authors introduce the Kernelized Movement Primitives (KMP) framework which they also extend to support multiple coordinate frames.

Not many approaches incorporate force information into the decomposition of the task (e.g., Gao et al. [31]). Abu-Dakka et al. [32] adapt the dynamic movement primitives (DMPs, Ijspeert et al. [33]) framework so that the robot follows a desired force profile. Yet, they predefine if an MP is position or force-controlled. Steinmetz et al. [34] learn a desired position and force profile with a DMP from a single demonstration. They present a control framework that is able to transition between phases of pure impedance control and force control based on the measured external force. The approach is not able to handle multiple demonstrations or sequences of MPs.

Ureche et al. [35] extract the coordinate frames and control variables based on the variance of the data. If the variance of a variable is large within a time window for all demonstrations, they consider it to be significant for the task. The reason is that the variable changes its value and does that in a systematic way across all demonstrations. Kober et al. [36] showed that under certain circumstances this assumption leads to an over-segmentation of the data. Therefore, they suggested incorporating the convergence behavior of the motion as well. Compared to our method, many approaches (e.g., Ureche et al. [35], Kober et al. [36], Pastor et al. [37], Mühlig et al. [38]) require demonstrations with identical sequential ordering of the employed MPs. Our approach is instead capable of decomposing a task if the demonstrated MP sequences vary across demonstrations. An example are demonstrations of a light bulb unscrewing task. Here, the number of unscrewing repetitions may vary over demonstrations depending on how firmly the light bulb is screwed in. Furthermore, our approach is able to simultaneously infer all possible MP sequences as well as the

composition of the MPs. We will show that decomposing kinesthetic demonstrations of a force interaction task with our approach leads to meaningful and intuitive MPs.

Many approaches focus on task-decomposition. While these approaches are capable of identifying reoccurring motions and representing them as MPs, they do not aim to learn how to sequence them for performing a complex task. One approach that learns to sequence MPs is that of Kappler et al. [39]. Here, the authors encode MPs as DMPs and link them with expected sensory data. Succeeding MPs are subsequently selected by comparing the current sensor values with the expected ones and choosing the best match. Other examples for sequence learning approaches are that of Ekvall and Kragic [40], Nicolescu and Mataric [41] or Pardowitz et al. [42] and some of the aforementioned decomposition approaches (e.g., Niekum et al. [25]). Medina and Billard [43] represent demonstrations with a set of asymptotically stable linear parameter varying systems and additionally learn the termination conditions between individual movements.

Compared to the presented related work in the field of learning force interaction skills, we do not concentrate on single learning aspects. Instead, our approach supports the whole pipeline from acquiring the demonstrations to reproducing the task on a real robot in a setting which is different from the demonstrations.

### *1.2. Used Controller and MP Framework*

In this paper, a MP corresponds to a single point-to-point movement and is represented by a dynamical system (DS) with a linear attractor behavior

$$\ddot{\boldsymbol{x}} = \alpha \left( \beta(\boldsymbol{x}_g - \boldsymbol{x}) - \dot{\boldsymbol{x}} \right), \tag{1}$$

where $\alpha$ and $\beta$ are controller parameters. As we aim at learning skills where the execution speed is not important, the controller parameters are predefined. Each MP has a goal $\boldsymbol{x}_g$ in task space coordinates $\boldsymbol{x}$ that should be reached if it is activated. Throughout this paper, a single MP will always control either the position/force or orientation of the end-effector (in world coordinates or relative to an object in the scene) or the joint angles of the robot's hand. For controlling the robot, we use a hybrid position-force controller based on task-level inverse dynamics. The controller as well as the used MP representation are for instance also used by Kober et al. [36]. The (desired) joint torque $\boldsymbol{T}$ is given by

$$\boldsymbol{T} = \boldsymbol{M}\boldsymbol{J}^{\#}\boldsymbol{S}\left(\boldsymbol{e}_x - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) + \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{S})\boldsymbol{e}_f + \boldsymbol{M}\boldsymbol{J}^{\#}(\boldsymbol{I} - \boldsymbol{S})\left(\boldsymbol{e}_d - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - \boldsymbol{M}\left(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}\right)\boldsymbol{\xi} + \boldsymbol{g} + \boldsymbol{h}. \tag{2}$$

An overview of the variables used in (2) can be found in Table 1. $\boldsymbol{J}$ is the task Jacobian and $\boldsymbol{J}^{\#}$ its pseudo-inverse. $I$ is the identity matrix, $q$ and $\dot{q}$ are the joint angles and joint angle velocities, respectively. Vector $\boldsymbol{\xi}$ accounts for joint speed damping and joint limit avoidance and is projected into the null space of the movement. $\boldsymbol{M}$, $\boldsymbol{h}$ and $\boldsymbol{g}$ denote the mass matrix, Coriolis forces and gravity, respectively. The MPs enter the equation via $\boldsymbol{e}_f$ and $\boldsymbol{e}_x$. Vector $\boldsymbol{e}_f$ contains the concatenated desired forces of the individual MPs. Vector $\boldsymbol{e}_x$ is the output of a PID controller which tracks the desired task-space accelerations of all MPs. If a MP is composed of task-variables controlling force and position of a robot body (e.g., Cartesian $x$ and $y$ position of end-effector and force along $z$ component in a certain coordinate frame), the individual variables are split up and assigned to $\boldsymbol{e}_f$ or $\boldsymbol{e}_x$ according to what they control. $\boldsymbol{S}$ is a diagonal selection matrix which enables selecting either kinematic or force components of a task variable and $\boldsymbol{e}_d$ is a task-level damping term. Based on the activations of the individual MPs, the pseudo-inverse is scaled with a weighting matrix. In theory, the weighting matrix allows for continuously modulating the contributions of the individual MPs, but in this paper a single MP is either fully active or deactivated. The desired paths of a deactivated MP are ignored by the controller.

**Table 1.** Overview of variables used by control framework.

| Variable | Description | Variable | Description |
|---|---|---|---|
| $\boldsymbol{I}$ | Identity matrix | $\boldsymbol{\xi}$ | (Desired) null-space coordinates |
| $\boldsymbol{J}$ | Task Jacobian | $\boldsymbol{e}_d$ | Task-level damping term |
| $\boldsymbol{J}^{\#}$ | Task Jacobian pseudo-inverse | $\boldsymbol{e}_f$ | (Desired) MP forces |
| $\dot{\boldsymbol{J}}$ | Task Jacobian time-derivative | $\boldsymbol{e}_x$ | Output of PID controller |
| $\boldsymbol{J}^{\mathrm{T}}$ | Task Jacobian transposed | $\boldsymbol{g}$ | Gravity vector |
| $\boldsymbol{M}$ | Mass matrix | $\boldsymbol{h}$ | Coriolis forces |
| $\boldsymbol{S}$ | Selection matrix | $\boldsymbol{q}$ | Joint angles |
| $\boldsymbol{T}$ | (Desired) joint torques | $\dot{\boldsymbol{q}}$ | Joint angle velocities |

### *1.3. Learning Sequential Force Interaction Tasks*

This section provides an overview of the presented method and the structure of the paper. Additionally, the notation that will be used throughout the paper is introduced. An overview of the approach is depicted in Figure 1. The aim of the approach is to learn to perform a sequential task from kinesthetic demonstrations. To learn the task, two main steps are performed. First, a set of MPs is extracted from the demonstrations. Second, the method learns how to sequence the resulting MPs.

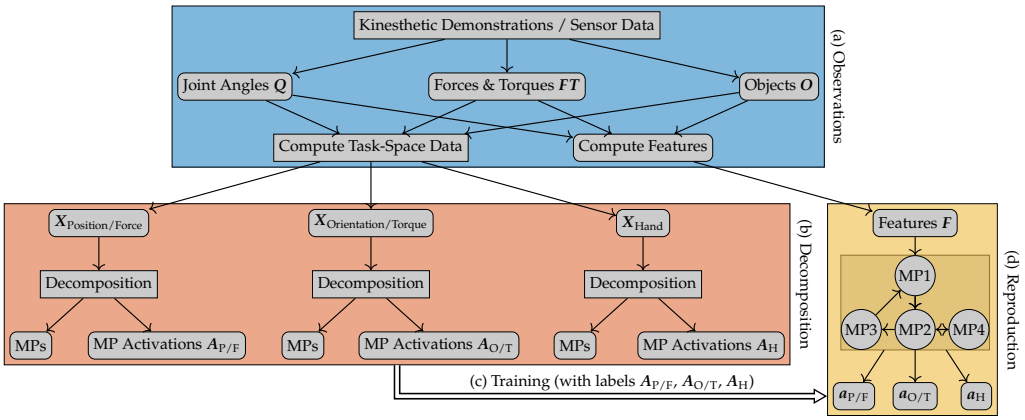



**Figure 1.** Overview of our approach. From the kinesthetic demonstrations (**a**), we record the joint angles of the robot, the measurements of the force-torque sensor and the positions and orientations of all objects in the scene. The observations are then projected onto a set of predefined task-spaces. The task-spaces are split according to what they control (**b**). The resulting data is used to decompose the demonstrated task, yielding a set of MPs and their activation probabilities $\boldsymbol{A}_{P/F}$, $\boldsymbol{A}_{O/T}$, $\boldsymbol{A}_H$ over time. When executing a task on a robot, the system controls the current movement by activating and deactivating the MPs found by the task-decomposition (**d**). To learn a mapping from the state of the world (represented by features $\boldsymbol{F}$) to the active MPs, we connect the MPs to a set of classifiers. Those classifiers are then trained to learn a mapping from the current feature state $\boldsymbol{f}$ to the MP activations for the next time step $\boldsymbol{a}_{P/F}$, $\boldsymbol{a}_{O/T}$, $\boldsymbol{a}_H$ (**c**). Please note that when reproducing the task on a real robot, the features do not come from the demonstrations but are directly computed from the current sensor data.

Our method operates on the data recorded during a set of kinesthetic demonstrations of the task. Each demonstration $i$ of varying length $N_i$ with a robot that has $N_j$ joints results in a time-series of joint angles $\boldsymbol{Q}^{(i)} \in \mathbb{R}^{N_j \times N_i}$, forces and torques $\boldsymbol{FT}^{(i)} \in \mathbb{R}^{6 \times N_i}$, as well as the 3D positions and 3D orientations of all $N_o$ objects in the scene $\boldsymbol{O}^{(i,o)} \in \mathbb{R}^{6 \times N_i}$. Here, $\boldsymbol{O}^{(i,o)}$ contains the positions and orientations of the $o$th object for demonstration $i$. Each object and the world are associated with a coordinate frame. The forces and torques are measured at the wrist of the robot. The joint angles include the joints of the robot arm and the joints of the robot's hand.

Subsequently, the demonstrations are projected onto a predefined set of task-spaces. For our tasks (as well as many industrial tasks), it is sufficient to control the position and orientation of the end-effector and the joint angles of the robot's hand. These three entities (position, orientation, fingers) can be controlled largely independently. Therefore, we define the following task-spaces: We use task-spaces representing the position and force of the end-effector in the world frame and relative to every object in the scene. To represent the orientation and torque of the end-effector, also task-spaces are used for the world frame and all other coordinate frames. The fingers of the robot are controlled in joint-space. Therefore, one additional task-space is used to describe the finger joints. The projection results in the task-space data $\boldsymbol{X}^{(i,k)}$. Here, the matrix $\boldsymbol{X}^{(i,k)} \in \mathbb{R}^{N_k \times N_i}$ corresponds to the data of the $i$th demonstration represented in the $k$th task-space. Please note that the number of dimensions $N_k$ can vary for the individual task-spaces. Our method is not restricted to these task-spaces, but they are sufficient for performing a wide range of tasks, as the experimental evaluation will show.

The proposed method analyzes the generated task-space data and decomposes the task into a set of MPs. For each MP, the method estimates the target coordinates and selects a coordinate frame and the control variables. The MP framework we use allows the activation of multiple MPs at the same time. It is, therefore, possible to activate MPs controlling the three entities independently. This co-activation greatly simplifies our MPs. For instance, we can reach different positions with a constant orientation by changing only the MP controlling the position of the end-effector. As we can activate the MPs controlling the three entities independently, the decomposition is also performed independently. The task-space data is therefore split into three distinctive sets according to what the task-spaces control (see Figure 1). Subsequently, the decomposition is performed on each set, resulting in a set of MPs and their activation probabilities over time for each controlled entity. The detailed explanation of our task-decomposition method can be found in the following Section 2. The core of the method is the Directional Normal Distribution, which is introduced in Section 3.

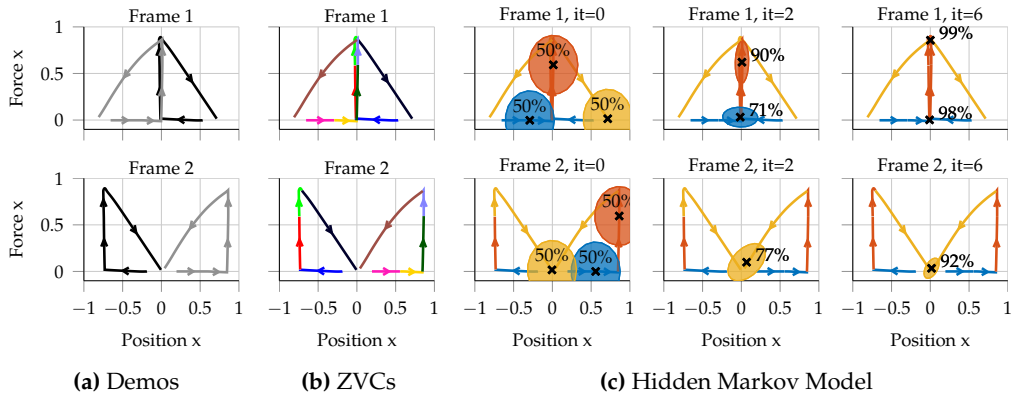
When reproducing a task on a real robot, the system has to decide when to activate which MP for each controlled entity (position, orientation, fingers). The decision which MPs to activate is made based on the current state of the robot and that of the environment, summarized in the feature set $\boldsymbol{f} \in \mathbb{R}^{N_F \times 1}$. Here, $N_F$ is the dimension of the feature set. To learn a mapping from features to MP activations, we record the features during the demonstrations $\boldsymbol{F}^{(i)} \in \mathbb{R}^{N_F \times N_i}$, as well. In general, the difference between the task-space vector and the feature vector is that the feature vector contains all the information that is necessary for deciding which movement to perform, while the task-space vector only represents the movements of the robot in relation to objects in the scene. Therefore, the feature vector may for instance also contain information about object sizes, color or texture if they are important for the task. However, in this paper, such an information is not needed and therefore, feature vector and task-space vector are equivalent. For more details on the sequencing method, the reader is referred to Section 4.

After learning to sequence the MPs, the task is reproduced on a real robot. For the current point in time, we compute the features and feed them into the MP sequencing method which then decides which MPs to activate. From the MP activations, we let the MPs compute the desired task-space goal and subsequently use (2) to compute the control command that is sent to the robot. In Section 5, we evaluate the proposed approach on three different tasks. Finally, we discuss the results and give a short outlook on future work in Section 6.

## 2. Proposed Task-Decomposition Approach

The task-decomposition approach consists of three steps. First, the demonstrations are split into smaller segments. Second, a model is trained which assigns the segments to individual MPs. Finally, the MPs are extracted from the parameters of the resulting model. The three steps will be explained in detail in the following subsections. The explanations are accompanied by a simple toy example that is using only position and force data. The reader can imagine the toy task as follows. The task is to first approach an object, subsequently push against the object and then move to a fixed final position.

Figure 2 depicts an overview of the toy example in the context of our task-decomposition method. As discussed earlier, the same method is used for the individual decompositions of the three controlled entities (position, orientation and fingers). However, handling orientations needs some adaptions that are explained in Section 3.2).




**(a)** Demos **(b)** ZVCs **(c)** Hidden Markov Model

**Figure 2.** Overview of the presented task-decomposition approach using a simple 1D toy example. Plot (**a**) shows two demonstrations (black and gray) in two different coordinate frames. Our approach first segments the data by finding zero-velocity crossings (ZVCs, Flanagan et al. [44]) and contact changes (**b**). The different segments are illustrated using different colors. Subsequently, the segments are clustered by using a Hidden Markov Model (HMM) (**c**). Here, we use mixtures of Directional Normal Distributions (DNDs) as state emissions of the HMM. DNDs allow for clustering the segments based on their convergence properties. If two segments are converging to the same attractor in one of the task-spaces corresponding to the coordinate frames, they are more likely assigned to the same cluster. After training, a MP is defined for each resulting cluster, and its coordinate frame, control variables and attractor target can be inferred from the parameters of the cluster. The attractor goals of the MPs are marked in the plot and the uncertainty about their position is shown with ellipsoids. The goals are only shown in the most likely coordinate frame of the MP. For instance, the yellow MP will be position-controlled in the second frame (with a certainty of 92%), as the target force is close to zero and the segments assigned to the MP converge best in this frame.

### *2.1. Segmentation*

As a first step, we split the demonstrations into smaller segments by finding the zero-velocity crossings (ZVCs) of the individual task-spaces. Hence, a segment is added if the robot gets into or looses contact with an object or if a controlled entity stopped and starts to move again. Flanagan et al. [44] showed the relevance of ZVCs for dexterous manipulation from a biological point of view. Detecting ZVCs is therefore widely used as preprocessing step for task-decomposition approaches (e.g., Kober et al. [36] or Lioutikov et al. [4]). After the segmentation step, the task-space data is split into sequences of segments $\boldsymbol{X}^{(i,k,l)} \in \mathbb{R}^{N_k \times N_l}$. Here, $N_l$ is the number of points of the $l$th segment of demonstration $i$, projected onto the $k$th task-space. Figure 2b depicts the segmentation of our toy example. After the segmentation, we assume the data is over-segmented, e.g., because the teacher stopped temporarily for some reason not related to the task. Hence, we usually have more segments than MPs and therefore have to assign the individual segments to MPs according to a similarity measure.

### *2.2. Assignment of Segments to MPs*

As we have assumed convergent attractor behavior of the MPs, we argue that segments converging to the same target (in at least one task-space) should be assigned to the same MP. The convergence property of the segments can be measured by a probability distribution which we call Directional Normal Distribution (DND). The distribution is needed as we assume our data is over-segmented. Therefore, we cannot simply cluster the end-points of the individual segments

in order to check if multiple segments can be represented by the same point-to-point movement. Instead, multiple segments may point towards a common attractor goal, even though their end-points may be far away from each other. Such an example is also illustrated in Figure 2. The DND ensures that we can analyze the convergence properties of the individual segments regardless of how close the end-points of the individual segments are.

This distribution is based on the normal distribution and has two parameters, a mean $\boldsymbol{\mu}$ and a Covariance matrix $\boldsymbol{\Sigma}$. The difference to a Normal Distribution is the meaning of the $\boldsymbol{\mu}$ parameter. While $\boldsymbol{\Sigma}$ can also be interpreted as an uncertainty parameter, the mean is defined as an attractor target. Hence, for a given segment the distribution yields a probability that indicates how well the segment is converging towards the attractor target. During learning, a DND is used for each MP and hence $\boldsymbol{\mu}$ corresponds to the most likely target of the MP. The learning algorithm will be explained in detail in Section 3. For this section, it is sufficient to know that given some segments, we can estimate the target of each MP in a probabilistic manner by using DNDs.

To find an assignment of segments to MPs, we use a Hidden Markov Model (HMM) to model the time-series sequence of segments. To incorporate the desired convergence property of the segments, we integrate the DND as state probability distribution in the HMM. As our data is represented in different task-spaces and we want to measure the convergence for each task-space and segment, we use a mixture of DNDs for each HMM state (and hence MP). One advantage of using an HMM is that we can perform model selection to choose the optimal number of MPs for the demonstrations. For this step, we use the Bayesian Information Criterion (BIC), a standard model selection criterion for probabilistic models.

### *2.3. Extraction of MP Parameters*

The HMM training assigns each segment a probability of being generated by one of the MPs. In Section 4, we show that this property can be used for learning to sequence the resulting MPs. In addition to the assigned probabilities, the parameters of the DND mixtures can be used to extract the MP parameters after the HMM training. For each MP, the largest mixture weight indicates in which of the task-spaces it is most likely represented (as it is converging best to its mean). The target of the MP is then defined s the mean of the mixture with the largest weight. Finally, if the assigned task-space of an MP is composed of forces and positions along the same axis, we explicitly decide whether force or position are chosen as the control variable. Here, we use a simple heuristic. Force is only chosen if the desired attractor force of the MP is higher than a threshold and the velocity mean of the segments assigned to this MP is below a threshold. Thus, if a force was measured along a certain axis and the robot was not moving in this direction, we assume the teacher wanted the robot to apply a force. The MP parameter extraction for the toy example is depicted in Figure 2c.

## 3. Measuring Convergence with the Directional Normal Distribution

We introduce the DND as an instrument for measuring the convergence of the individual segments. The distribution has two parameters $\theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. The mean $\boldsymbol{\mu}$ corresponds to an attractor target, while the covariance matrix $\boldsymbol{\Sigma}$ indicates the uncertainty about the location of $\boldsymbol{\mu}$. To be used for our purpose, the distribution has to fulfill two properties. First, given a segment $\boldsymbol{X}$ the probability $p(\boldsymbol{X}|\theta)$ should be large if the segment is converging to the target $\boldsymbol{\mu}$, and small otherwise. Second, given a single or multiple segments, we need a method for estimating the most likely parameters of the distribution. In this section, we present the idea behind the distribution and its density function. In Section 3.1, we then derive a method for estimating the parameters from data using a maximum likelihood approach.

For deriving the density function, we assume the following statement to hold. If a segment is converging to a given target, then on average, the velocity vector $\boldsymbol{v} \in \mathbb{R}^{d\times 1}$ corresponding to a data point $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$ (in our context $d$ is the dimension of the task-space vector) from the segment should

roughly point towards the target. Thus, the probability of a segment is computed as joint probability of the points from the segment

$$p(\boldsymbol{X}|\theta) = \prod_{m=1}^{N} p\left([\boldsymbol{x}_m, \boldsymbol{v}_m]\,|\theta\right)$$

Here, $N$ is the number of points of the segment and $\boldsymbol{x}_m$ and $\boldsymbol{v}_m$ are the individual points from the segment. Taking into account the convergence property of an entire segment allows for inferring if a segment is converging to an attractor, even though the end-point of the segment may be far away from the attractor. Given a single data point $\boldsymbol{x}$ and its velocity vector $\boldsymbol{v}$, the density function of a DND is defined as

$$p\left([\boldsymbol{x}, \boldsymbol{v}]\,|t, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{e^{-\frac{1}{2}(\boldsymbol{x}+t\boldsymbol{v}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}+t\boldsymbol{v}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^d\,|\boldsymbol{\Sigma}|}}. \tag{3}$$

Compared to a normal distribution, the DND also incorporates the direction of a data point, given by the velocity vector $\boldsymbol{v}$. The scalar parameter $t \geq 0$ projects the point along its velocity vector. The idea is to choose the parameter so that the distance between the projection and the mean of the distribution becomes minimal. Hence, if a point $\boldsymbol{x}$ is moved along $\boldsymbol{v}$ for the duration $t$, it reaches the minimum distance to $\boldsymbol{\mu}$, under consideration of the uncertainty $\boldsymbol{\Sigma}$. By rearranging (3), it can also be written as a normal distribution of $t$

$$p\left(t|\mu_t, \sigma_t^2\right) = Z_c e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}, \tag{4}$$

with normalization constant $Z_c$. Due to the constraint $t \geq 0$, we can assume that the posterior of $t$ is distributed to a truncated normal distribution

$$\begin{aligned} p\left(t|\,[\boldsymbol{x}, \boldsymbol{v}]\,, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) &= \frac{1}{\sqrt{(2\pi)}\sigma_t}\frac{1}{1-\Phi(-\frac{\mu_t}{\sigma_t})}e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}, \\ \mu_t &= \frac{\boldsymbol{v}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{x})}{\boldsymbol{v}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{v}}, \\ \sigma_t &= (\boldsymbol{v}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{v})^{-\frac{1}{2}}. \end{aligned} \tag{5}$$

The equations for the variables $\mu_t$ and $\sigma_t$ result from rearranging the exponential term in (3) such that it matches that of (5). In (5), $\Phi$ is the cumulative distribution function of a normal distribution with standard deviation one and mean zero. To avoid running into numerical issues, we set $\mu_t$ to zero and $\sigma_t$ to one if the norm of the velocity is below a threshold. We would also like to point out that the projection parameter $t$ serves as a normalization constant for the velocity. For smaller velocities, the parameter will be larger, as only the direction of the velocity is considered. This property is beneficial for our purpose, as we are not interested in the velocity of a movement and do not have to preprocess the velocities. The joint probability of the data point and its projection is defined as

$$p\left([\boldsymbol{x}, \boldsymbol{v}]\,, t|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = p\left([\boldsymbol{x}, \boldsymbol{v}]\,|t, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) p\left(t|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right). \tag{6}$$

where we omitted the dependency of the joint probability on the parameters of the distribution. When learning the parameters of the distribution, we are not interested in the projection itself. Instead, we want to integrate it out, as it depends on the (unknown) parameters of the distribution. Given data $\boldsymbol{X}$ and $\boldsymbol{V}$, the log-likelihood function is defined as

$$\log p\left([\boldsymbol{X}, \boldsymbol{V}] \,|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \sum_{i=1}^{N} \int_0^{\infty} \log p\left([\boldsymbol{x}_i, \boldsymbol{v}_i], t_i | \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) dt_i.$$

Here, we assumed that the values $t_i$ are independent of each other. We justify this assumption by the fact that given a fixed $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the $t_i$'s can be computed independently of each other. We would also like to point out that the individual points $\boldsymbol{x}_i$ all come from a continuous trajectory with a continuously changing velocity vector $\boldsymbol{v}_i$. Therefore, the projections $\boldsymbol{x}_i + t_i \boldsymbol{v}_i$ will automatically be close nearby for succeeding data points. To infer the parameters of our distribution, we want to maximize the log-likelihood function. The required integration over $t_i$ is intractable as the variables depend on the unknown parameters $\mu_t^{(i)}$ and $\sigma_t^{(i)}$. Therefore, we derive an Expectation-Maximization algorithm (EM) to get an iterative update scheme for the parameters.

### *3.1. Parameter Learning*

In the context of the task-decomposition, the aim is to find the best attractor target for a given set of segments when learning the parameters of the distribution. As the probability is defined as the joint probability of the individual points from the segments, it does not matter if the parameters are learned for a single or multiple segments. Therefore, the notion of a segment is omitted in this section.

The principle of the EM algorithm is depicted in Figure 3. In the Expectation step of the algorithm, we use the current parameter values $\theta^{\text{old}} = \{\boldsymbol{\mu}^{\text{old}}, \boldsymbol{\Sigma}^{\text{old}}\}$ to estimate the posterior distribution of the latent variables $p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{V}, \boldsymbol{\theta}^{\text{old}})$. Here, $\boldsymbol{t} = [t_1, t_2, \ldots, t_N]^T$ is the concatenation of all projections. As the $t_i$'s are independent of each other, the estimation can be done for each data point separately as in (5). Intuitively, the E-step projects each data point as close as possible to the current mean of the distribution. In the Maximization step, we get new estimates for our parameters $\boldsymbol{\theta}^{\text{new}}$ by maximizing the expectation of the complete-data log-likelihood under the posterior of the latent variables with respect of the parameters $\boldsymbol{\theta}$

$$\begin{aligned}\boldsymbol{\theta}^{\text{new}} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{t}}\left[\log p\left([\boldsymbol{X}, \boldsymbol{V}], \boldsymbol{t} | \boldsymbol{\theta}, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\right)\right] \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \int_0^{\infty} p\left(t_i | \left[\boldsymbol{x}_i, \boldsymbol{v}_i\right], \boldsymbol{\theta}\right) \log p\left(\left[\boldsymbol{x}_i, \boldsymbol{v}_i\right], t_i | \boldsymbol{\theta}, \mu_t^{(i)}, \sigma_t^{2(i)}\right) dt_i. \end{aligned} \tag{7}$$

As the integral is evaluated for each data point separately, we omit the indices for the solution of the integral in the following. The integral evaluates to

$$\begin{aligned}\int_0^{\infty} p\left(t | \left[\boldsymbol{x}, \boldsymbol{v}\right], \boldsymbol{\theta}\right) \log p\left(\left[\boldsymbol{x}, \boldsymbol{v}\right], t | \boldsymbol{\theta}, \mu_t, \sigma_t\right) dt &= \int_0^{\infty} c_1 e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}} \left(c_2(\boldsymbol{\theta}) t^2 + c_3(\boldsymbol{\theta}) t + c_4(\boldsymbol{\theta})\right) dt \\ &= c_1 \left(c_2(\boldsymbol{\theta}) d_1 + c_3(\boldsymbol{\theta}) d_2 + c_4(\boldsymbol{\theta}) d_3\right). \end{aligned} \tag{8}$$

The values of the constants can be found in Appendix A. Please note that $c_2, c_3$, and $c_4$ are constants only for the integration, as they depend on the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ which we want to estimate. Upon evaluating this integral, the parameters can be obtained using (7). As the constants $c_2$ to $c_4$ depend on the parameters, we compute the derivatives of (8) with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ and set them to zero. Now we work again on the entire data set, so we will use the indices again.

$$\begin{aligned}\frac{\partial \mathbb{E}_{\boldsymbol{t}}\left[\log p\left([\boldsymbol{X}, \boldsymbol{V}], \boldsymbol{t} | \boldsymbol{\theta}, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\right)\right]}{\partial \boldsymbol{\mu}} &= \sum_{i=1}^{N} c_1^{(i)} \left(d_1^{(i)} \frac{\partial c_2^{(i)}}{\partial \boldsymbol{\mu}} + d_2^{(i)} \frac{\partial c_3^{(i)}}{\partial \boldsymbol{\mu}} + d_3^{(i)} \frac{\partial c_4^{(i)}}{\partial \boldsymbol{\mu}}\right) \\ &= \boldsymbol{\Sigma}^{-1} \sum_{i=1}^{N} c_1^{(i)} \left(d_2^{(i)} \boldsymbol{v}_i - d_3^{(i)} \boldsymbol{\mu} + d_3^{(i)} \boldsymbol{x}_i\right) = \boldsymbol{0}. \end{aligned}$$

Multiplying by $\boldsymbol{\Sigma}$ from the left and rearranging leads to the solution

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{N} c_1^{(i)} d_3^{(i)} \boldsymbol{x}_i + c_1^{(i)} d_2^{(i)} \boldsymbol{v}_i}{\sum_{i=1}^{N} c_1^{(i)} d_3^{(i)}}. \tag{9}$$

Please note that the cumulative distribution function $\Phi$ is closely related to the error function erf by the relation

$$\Phi\left(-\frac{\mu_t}{\sigma_t}\right) = \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right).$$

Therefore, the constants $c_1^{(i)} d_2^{(i)}$ and $c_1^{(i)} d_3^{(i)}$ could be further simplified. For instance, $c_1^{(i)} d_3^{(i)}$ equals one (proof skipped here) and so (9) can also be written as

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{x}_i + c_1^{(i)} d_2^{(i)} \boldsymbol{v}_i\right).$$

For estimating the covariance matrix $\boldsymbol{\Sigma}$, we compute the derivative with respect to its inverse $\boldsymbol{\Sigma}^{-1}$.

$$\begin{aligned}
\frac{\partial \mathbb{E}_{\boldsymbol{t}}\left[\log p\left([\boldsymbol{X}, \boldsymbol{V}], \boldsymbol{t} \middle| \boldsymbol{\theta}, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\right)\right]}{\partial \boldsymbol{\Sigma}^{-1}} &= \sum_{i=1}^{N} c_1^{(i)} \left(d_1^{(i)} \frac{\partial c_2^{(i)}}{\partial \boldsymbol{\Sigma}^{-1}} + d_2^{(i)} \frac{\partial c_3^{(i)}}{\partial \boldsymbol{\Sigma}^{-1}} + d_3^{(i)} \frac{\partial c_4^{(i)}}{\partial \boldsymbol{\Sigma}^{-1}}\right) \\
&= \frac{1}{2} \sum_{i=1}^{N} c_1^{(i)} \Big( - d_1^{(i)} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} + d_2^{(i)} \boldsymbol{v}_i (\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} \\
&\quad + d_2^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i) \boldsymbol{v}_i^{\mathrm{T}} - d_3^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i)(\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} + d_3^{(i)} \boldsymbol{\Sigma} \Big) = \boldsymbol{0}.
\end{aligned}$$

Rearranging then leads to the solution

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} c_1^{(i)} \left(d_1^{(i)} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} - d_2^{(i)} \boldsymbol{v}_i (\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} - d_2^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i) \boldsymbol{v}_i^{\mathrm{T}} + d_3^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i)(\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}}\right). \tag{10}$$

In summary, the parameters of the distribution can be estimated iteratively by first computing the $c$ and $d$ constants for each data point according to Appendix A (E-step). Second, a new estimate for the parameters of the distribution can be found by computing $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ according to (9) and (10) (M-step). The algorithm is summarized in Algorithm 1.

**Algorithm 1** EM-algorithm for DND

**Require:** data $\boldsymbol{X}$, velocities $\boldsymbol{V}$, initial mean $\boldsymbol{\mu}$, initial cov $\boldsymbol{\Sigma}$

**while** not converged **do**
  **for all** data points $\boldsymbol{x}_i$, $\boldsymbol{v}_i$ **do** ▷ Start E-step
    compute constants $c_1^{(i)}$ and $d_1^{(i)}, \ldots, d_3^{(i)}$ (Appendix A)
  **end for**
  compute new mean $\boldsymbol{\mu}(\boldsymbol{c}_1, \boldsymbol{d}_3, \boldsymbol{d}_4)$ (9) ▷ Start M-step
  compute new covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{c}_1, \boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3)$ (10)
**end while**
**return** mean $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$

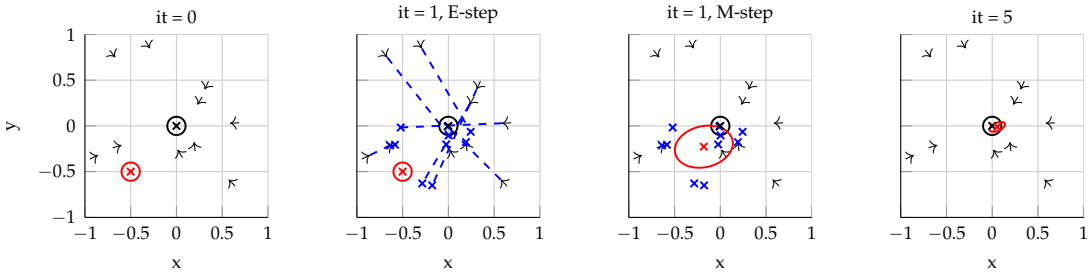



**Figure 3.** Illustration of the EM algorithm. The black arrows correspond to data points and their velocity vectors. The data points are drawn uniformly from the shown grid. The velocity vectors are determined by drawing points from the black normal distribution and scaling the difference vector to the data points to a fixed value. The EM algorithm iteratively finds a target on the grid that fits best to all data points and the corresponding velocity vectors. The red ellipsoid shows the initial and current guesses for the target. In the E-step of the algorithm, for each data point a projection along the velocity vector is computed that fits best to the current estimate of the target (blue dashed lines). In the M-step, a new estimate for the distribution parameter is found.

### *3.2. Extension for Orientations*

The Euclidean distance measure used in (3) is not suitable for 3D orientations. In this section, we show that by making some reasonable approximations, we can make our algorithm applicable also for orientations.

To be applicable, we have to define a projection $\boldsymbol{x} + t\boldsymbol{v}$ and a difference vector for orientations that can be written in the form $\boldsymbol{x} + t\boldsymbol{v} - \boldsymbol{\mu}$. We define the target orientation in the inertia frame with the rotation matrix $\boldsymbol{R}_{\mathrm{FI}}$. The matrix $\boldsymbol{R}_{\mathrm{VI}}(t) = \boldsymbol{R}_{\mathrm{VS}}(t)\boldsymbol{R}_{\mathrm{SI}}$ represents the start orientation rotated with the angle $t$ around the angular velocity axis. The rotation matrix is also given in the inertia frame. Hence, the projection parameter $t$ has a different meaning than for Euclidean task-spaces. Instead of projecting a position along its velocity vector, it rotates an orientation by an angle around the angular velocity axis. By writing the matrices as vectors $\boldsymbol{r} \in \mathbb{R}^{9\times 1}$, we can define a difference vector with

$$\begin{aligned}
\boldsymbol{r}_{\mathrm{VI}}(t) - \boldsymbol{r}_{\mathrm{FI}} &= \boldsymbol{d}\cos(t) + \boldsymbol{e}\sin(t) + \boldsymbol{f} - \boldsymbol{r}_{\mathrm{FI}} \\
&\approx \boldsymbol{d}(\cos(\alpha) - \sin(\alpha)(t-\alpha)) + \boldsymbol{e}(\sin(\alpha) + \cos(\alpha)(t-\alpha)) + \boldsymbol{f} - \boldsymbol{r}_{\mathrm{FI}}, \\
&= \boldsymbol{x} + t\boldsymbol{v} - \boldsymbol{\mu}, \\
\boldsymbol{x} &= \alpha\cos(\alpha)(-\boldsymbol{d} - \boldsymbol{e}) + \alpha\sin(\alpha)(-\boldsymbol{d} + \boldsymbol{e}) + \boldsymbol{f}, \\
\boldsymbol{v} &= \cos(\alpha)(d + e) + \sin(\alpha)(-d + e), \\
\boldsymbol{\mu} &= \boldsymbol{r}_{\mathrm{FI}}.
\end{aligned} \tag{11}$$

Please note that the constants $\boldsymbol{d}$, $\boldsymbol{e}$ and $\boldsymbol{f}$ are different from those derived before. The derivation can be found in Appendix B. Now our difference vector can be written in the form $\boldsymbol{x} + t\boldsymbol{v} - \boldsymbol{\mu}$ and we can estimate the target orientation with the same algorithm as in the previous section. Still, some details have to be taken care of. First, we approximated the sine and cosine terms with a first-order Taylor approximation around $\alpha$. We therefore have to evaluate how good the approximation is. Second, the approximation introduced the parameter $\alpha$ and it is not clear how to choose it. Third, estimating the parameters of the distribution with our standard algorithm will result in a value for $\boldsymbol{\mu}$ which does not describe a proper orientation. Therefore, after applying our standard decomposition method, we reshape the target vector to a matrix and find the closest true rotation matrix to our matrix using [45].

In the following, we introduce an intuitive way of choosing $\alpha$. A natural way of measuring the difference between two orientations is the axis angle $\theta(t)$ of the relative transformation $\boldsymbol{R}_{\mathrm{FV}} = \boldsymbol{R}_{\mathrm{FI}}(\boldsymbol{R}_{\mathrm{VI}})^{\mathrm{T}}$

$$\begin{aligned}\theta(t) &= \cos^{-1}\left(\frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{R}_{\mathrm{FV}}) - 1\right)\right) \\ &= \cos^{-1}\left(\frac{1}{2}\left(a\ \cos(t) + b\sin(t) + c - 1\right)\right).\end{aligned} \tag{12}$$

The constants $a, b$ and $c$ are again different from those derived before, For the derivation, see Appendix B.1. To project the current orientation as closely as possible to the target orientation, we compute the derivative of (12) with respect to $t$ and set it to zero

$$\begin{aligned}\frac{\partial\theta}{\partial t} &= \frac{0.5a\ \sin(t) - 0.5b\cos(t)}{\sqrt{1 - \left(\frac{1}{2}\left(a\ \cos(t) + b\sin(t) + c - 1\right)\right)^2}} \\ &= 0 \quad \Rightarrow \quad t = \mathrm{atan2}(b, a).\end{aligned} \tag{13}$$

Hence, if we knew the target orientation without any uncertainty, we could compute a distance measure between the current orientation and the target in closed form. For the Taylor approximation, we can therefore make use of a trick. If we neglect the uncertainty $\boldsymbol{\Sigma}$, we can set the target orientation to the current estimate of the mean $\boldsymbol{\mu}$. Subsequently, we compute $\tilde{t}$ according to (13) and use it for the Taylor approximation $\alpha = \tilde{t}$. Therefore, we get a good approximation by developing the Taylor series about a value close to the true $t$. The approximation allows us to integrate orientations in our method in a straightforward way. Please note that we neglect the uncertainty about the target orientation only for the projection of the current orientation towards the current estimate of the target orientation. However, we still get an estimate of the uncertainty $\boldsymbol{\Sigma}$ about the target rotation $\boldsymbol{\mu}$. We consider it future work to also integrate the uncertainty into the projection.

## 4. Movement Primitive Sequence Learning

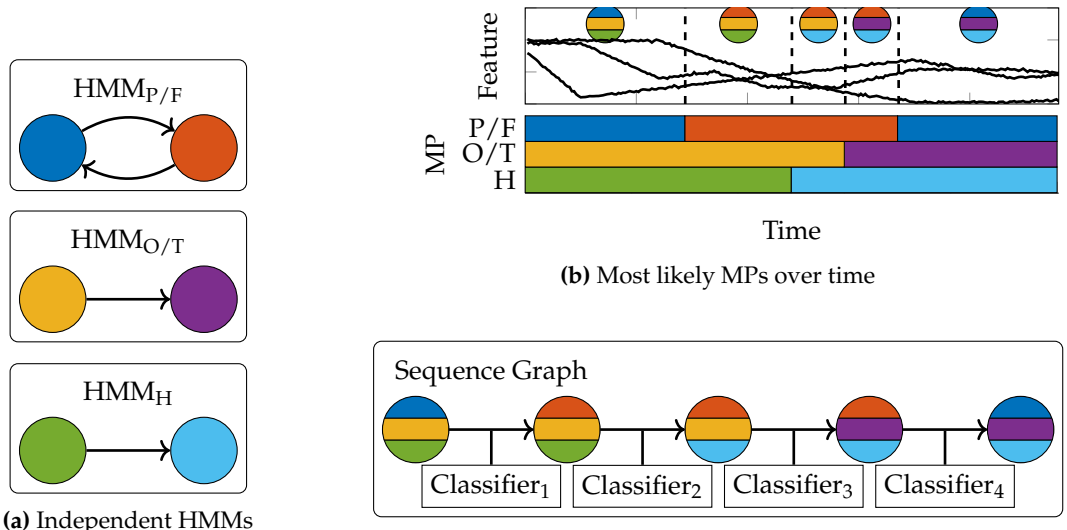
The overall aim of our approach is learning a skill that has a reactive behavior. Instead of following a pre-computed path, the system is supposed to decide online when to stop a movement and which movement to start next. For a successful execution of a learned skill, the MPs have to be activated at the correct time points and in the correct order. The task-decomposition presented in the previous sections can be used for inferring when the teacher performed which point-to-point movement, but it is not a decision-making system which learns a reactive behavior. Hence, the task-decomposition method allows for inferring *when* the teacher performed *which* movement, but not *why*. For answering this question, we use the sequence graph concept which was initially presented by Manschitz et al. [2,3]. In a sequence graph, a node corresponds to a MP. Each node is also associated with a local classifier. When executing a skill, one node in the graph is considered active and the associated classifier decides when to transition to a succeeding node, leading to the activation of another MP.

Applying our task-decomposition method to the demonstrations results in three independent sets of MPs, as the MPs controlling the position/force of the end-effector, its orientation and the joint angles of the fingers are learned independently of each other. For each set, we also get a HMM which reflects the potential sequential orderings of the MPs. While extracting the MPs of the three controlled entities independently of each other simplifies learning and leads to simple reusable MP, the MPs cannot be treated independently of each other when executing a skill on a real robot. For instance, the end-effector may only be allowed to move after an object has been grasped. Therefore, the decision when to execute which MP for one entity has to be conditioned on the state of the two other entities.

To account for the necessary synchronization of the MPs, we adapt the sequence graph concept for our purpose. Our system has to decide when to activate which of the MPs based on the current

state of the robot and the state of the environment (e.g., current positions of objects in the scene). This decision has to be made based on the feature state $\boldsymbol{f} \in \mathbb{R}^{N_f \times 1}$, where $N_f$ is the dimension of the feature space. In theory, this feature state can be comprised of arbitrary values (e.g., camera input), but in this paper, it will be equivalent to the task-space data $\boldsymbol{x}$. Thus, it implicitly encodes the state of the robot in relation to all objects in the scene (including the forces) and contains the joint angles of the fingers as well as the measured forces.

To synchronize the activations of the MPs of the three different entities, we use a single sequence graph where the active node activates three MPs, one for each controlled entity. This process is depicted in Figure 4. First, we use the HMMs resulting from the task-decomposition for labeling the demonstrations with the most likely MPs over time. From these labels, we generate the sequence graph representing the sequential structure of the MPs. Finally, we train the local classifiers for each node, where a classifier maps the feature state to three activation vectors $\boldsymbol{f} \rightarrow \{\boldsymbol{a}_{\mathrm{P/F}}, \boldsymbol{a}_{\mathrm{O/T}}, \boldsymbol{a}_{\mathrm{H}}\}$, one for each controlled entity. Therefore, when executing a learned skill on a robot, a transition leads to the activation of (up to) three new MPs. To train a classifier, each class is associated with the feature data that was recorded when the corresponding MP was considered active. For instance, the first classifier of the toy example in Figure 4 has to decide when to transition to the second node in the graph. The feature data before the first dashed line in the feature plot is assigned to the preceding node, while the data between the first and second dashed line are assigned to the succeeding node. Based on these data, the classifier learns to discriminate both nodes. When executing the skill on the robot and the features reach a state which the classifier assigns to the succeeding node, the system will switch to this node and change the currently active MP. We use Logistic Regression for training, but in general any classifier can be used. Please note that there are also alternative more sophisticated ways to synchronize the sequences (e.g., generate three graphs and couple their transitions), but finding the best way to synchronize the controlled entities is beyond the scope of this paper.




**(a)** Independent HMMs

**(b)** Most likely MPs over time

**(c)** Combined Sequence Graph

**Figure 4.** Overview of the sequence learning step. The three HMMs (**a**) resulting from the task-decomposition are used for labeling the demonstrations with the most likely MPs (different colors) over time (**b**). The labeled demonstrations are compiled into a single sequence graph representing potential MP orders (**c**). Local classifiers are responsible for transitioning between successive MPs when executing the skill on a real robot. The classifiers learns to discriminate the individual classes in feature space (state of robot and the world). The dashed lines indicate transition points between the classes, where a transition is supposed to be triggered. Please note that the features in the plot are only illustrative and have no further meaning.

## 5. Evaluation of the Approach

For evaluating our approach, we performed kinesthetic demonstrations on a gravity compensated seven degree of freedom (DOF) Barrett WAM robot with a four DOF hand. We performed three different tasks: box flipping, box stacking, and unscrewing a light bulb. For all tasks, we evaluate the decomposition and reproduction of the task. The purpose of the first task is to evaluate if our method is able to distinguish between position and force control. For the box stacking task, the focus is on finding the correct coordinate frames. The light bulb unscrewing task is a longer and more complex task which requires both, i.e., motions relative to multiple objects as well as force and position control. For the box stacking task, we also compare our decomposition method to a baseline method and two state-of-the-art methods.

All demonstrations for the experiments were performed by the same teacher which was one of the authors. The target of this paper is to evaluate whether our method can in principle be used for learning sequential skills from human demonstrations. We consider it future work to train the system on data from more users with different levels of experience. Please also note that it is hard to define evaluation metrics for learning from demonstration methods such as ours. Often, a system either can perform a task (resulting in 100% success rate for this task) or fails completely. Therefore, we rather provide a qualitative comparison between our method and other state-of-the-art methods.

For simplicity, we predefined the MPs controlling the fingers of the robot for all tasks, despite that teaching grasps is possible with our approach. During the demonstrations, we activated these MPs manually by pressing a key on a keyboard. Please note that we predefined the MPs only for simplifying the teaching process. We still applied our task-decomposition to the resulting finger joint angles data. The kinesthetic demonstration data was recorded with 200Hz. As the force-torque measurements are quite noisy, we filtered the data using a Hann filter with a window size of 100. Due to the noisy force sensor and the different units (Newton vs. meters), the force data was additionally scaled by the factor $1/40$, so that 4 N difference correspond to a position difference of 10 cm. For all experiments, force was chosen as the control variable for a dimension of a task-space if the target force was above the threshold of 2.5 N and the effector was not moving. An effector was considered to be not moving if the average velocity was below the threshold of $10^{-3}$ m/s (see Section 2.3).

### *5.1. Box Flipping*

For the box flipping task, the sequence of subtasks demonstrated by the teacher is shown in Figure 5. To flip the box, the teacher first pushed it against the obstacle. Then, he flipped it by pushing the box against the obstacle while rotating it. We demonstrated the task five times with slightly varying starting positions of the box. Two coordinate frames were defined for the task-decomposition, relative to the obstacle. A Cartesian frame and a cylindrical coordinate frame whose axis was aligned with the long side of the obstacle, as shown in Figure 6. Kober et al. [36] evaluated their approach on a similar box flipping task. They also used a cylindrical coordinate frame because it works very well for rotatory movements such as opening a door. Additionally, they stated that it is very robust with respect to inaccuracies in the rotation axis which is important when using force control. In contrast to their approach, we did not have to align the demonstrations in time to decompose the task. The orientation of the end-effector and the fingers were held constant throughout the demonstrations and are therefore neglected here.

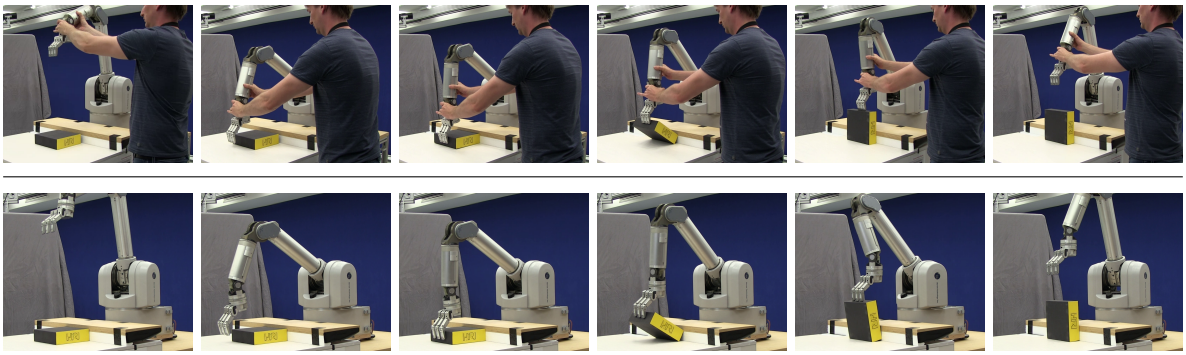


**Figure 5.** Illustration of teaching and reproduction of the box flipping task. The task was to initially move the end-effector to a position close to the box. Subsequently, the box was pushed against the obstacle. Then, the box was flipped. Finally, the end-effector was moved to its final position.

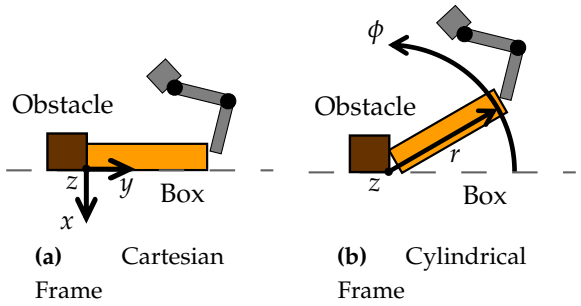
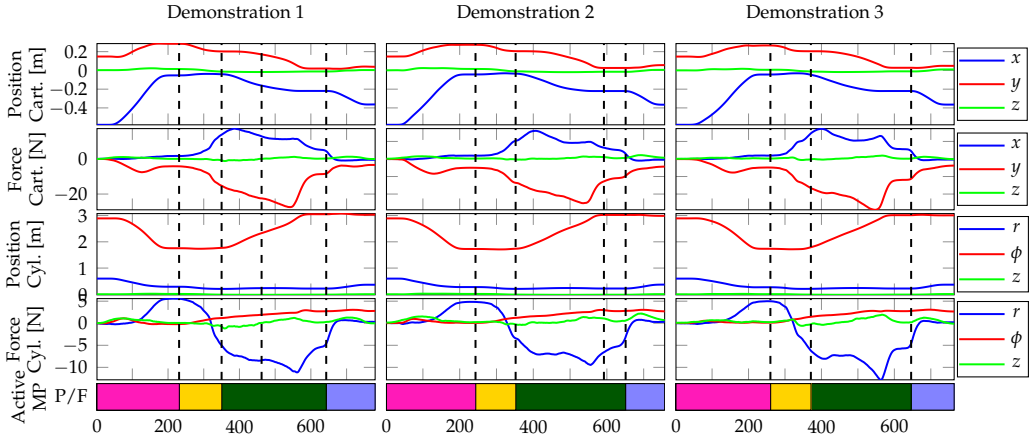
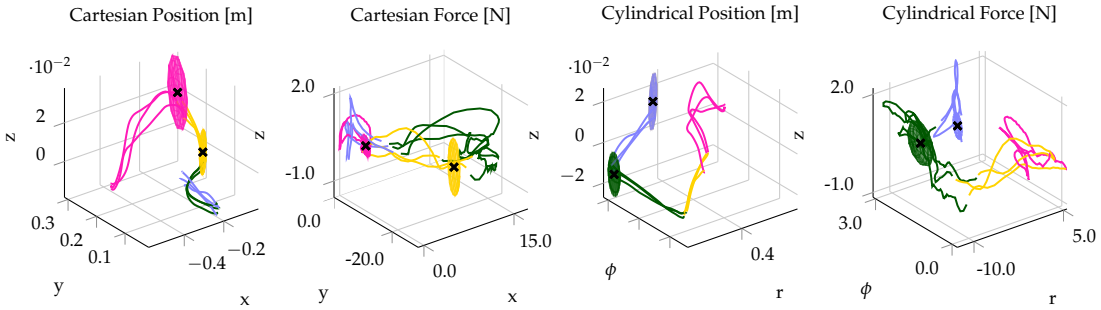



**Figure 6.** The two coordinate frames of the box flipping task shown from the side. The origin of the Cartesian frame is at the lower edge of the obstacle. The $z$-axes of both frames are identical and correspond to the lower edge of the obstacle (indicated by a black dot). Here, the origin is at the center of the lower edge. Please note that the frames are attached to the obstacle and not to the box.

Our task-decomposition algorithm resulted in four MPs. Three of the five demonstrations and the corresponding results are depicted in Figure 7. The results show that the decomposition is consistent over all demonstrations, even though the demonstrations are slightly over-segmented. The MPs closely resemble the behavior of the teacher (see Table 2). Initially, the ●-MP approaches the box. Subsequently, the ●-MP pushes the object against the box. Then, the ●-MP flips the box and the ●-MP moves the end-effector to its final position. For the first two MPs, the algorithm chose the Cartesian frame, while for the latter two, it chose the Cylindrical frame. The frame choices can be better understood when looking at the demonstrations plotted spatially (see Figure 7b). The box flipping resulted in a (green) curved line in Cartesian space and a straight line in the cylindrical space, which is why the algorithm choose the cylindrical frame for the ●-MP. For the remainder of the task, the teacher tried to move the end-effector in a straight line. Still, the algorithm chose the cylindrical frame for the final ●-MP. As shown in the plots, the final position slightly varied over the demonstrations and the teacher did not really move the end-effector in a straight line. The low confidence of 51.5% shows that the demonstrations seem to converge slightly better to a desired attractor target in the Cylindrical coordinate frame.




**(a)** Demonstrations and task-decomposition over time.




**(b)** Demonstrations and task-decomposition in 3*D* space.

**Figure 7.** Experimental results for the flipping task. The MP plot shows the active MP in different colors. It can be seen from the plots that the decomposition of the task is consistent throughout all demonstrations.

**Table 2.** Resulting description, coordinate frame and force selection for each MP of the box flipping task. Only the MP that is active when flipping the box is applying a force along the radial axis of the cylindrical coordinate frame. The confidence corresponds to the mixture weight of the winner frame.

| MP | Description | Frame | Confidence | Force |
|---|---|---|---|---|
| ● | Approach Box | Cartesian | 95.2% | - |
| ● | Push Box | Cartesian | 58.3% | - |
| ● | Flip Box | Cylindrical | 98.4% | Radial |
| ● | Approach Final Position | Cylindrical | 51.5% | - |

Only for the box flipping MP (●), the algorithm chose force control for one dimension of the coordinate frame. The reason is that while this MP was active, the radial distance to the origin of the coordinate frame was constant and hence the corresponding task coordinates did not change. As the target force of the MP along the radial axis of the coordinate frame exceeded the predefined threshold of 2.5 N and no velocity was measured, force control was chosen for this dimension of the task-space.

After decomposing the task and learning to sequence the MPs, the learned skill was successfully reproduced on the real robot. The found MPs were sufficient to perform the necessary movements and the transitions between MPs were triggered at the correct states. For the sequence learning, the following features were used: The end-effector position in the cylindrical frame and the Cartesian frame as well as the data from the force sensor at the wrist of the robot.

### *5.2. Box Stacking*

The goal of the second task is to stack four boxes on top of each other. Position and orientation of the boxes are tracked with Augmented Reality (AR) markers using the ArUco library [46]. We performed three demonstrations of the task. For each demonstration, the initial positions and orientations of the boxes were chosen randomly. The stacking sequence was the same for all demonstrations. First, the green box was put on the yellow box. Then, the red box was put on the green box and finally, the blue box was put on the red box. Figure 8 shows a demonstration of the task. For a successful stacking of the boxes, it is important that all MPs use the correct coordinate frames. The task does not require any force control and therefore the force data is omitted here.

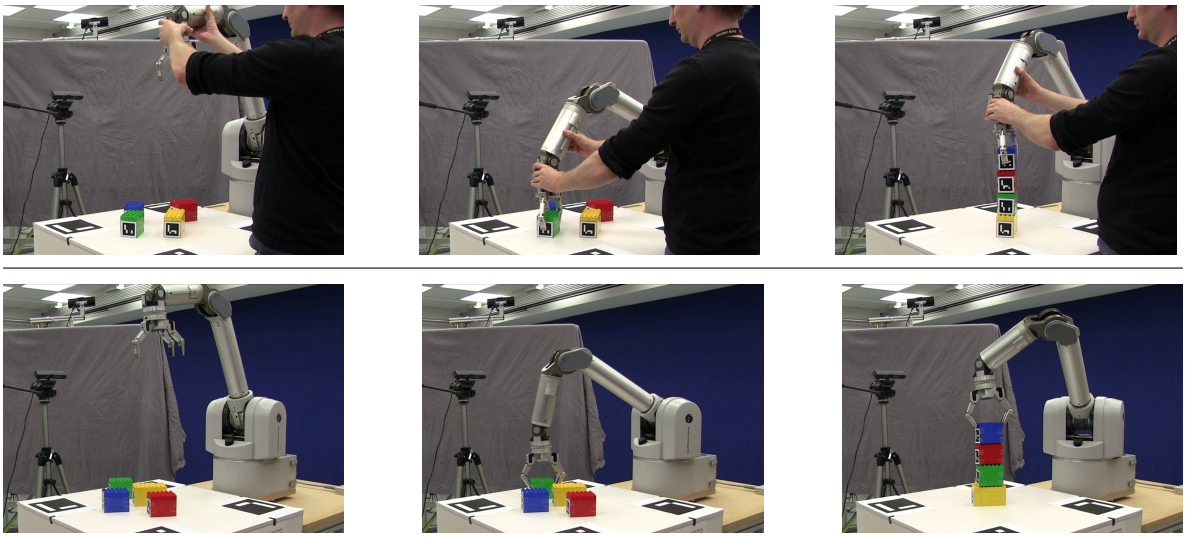


**Figure 8.** Experimental setup for the box stacking task (**top**) and pictures from the reproduction (**bottom**). The four boxes were put to random initial positions for each demonstration and the reproduction. They were tracked using AR markers.

As an example, Figure 9 depicts the data of one demonstration and the corresponding sequence of most likely MP activations. In total, our algorithm extracted seven position MPs, four orientation MPs and the two hand MPs corresponding to the ones we defined for grasping and opening the fingers. The MPs can be described as follows. In the beginning, the end-effector approaches the green box (●) with opened fingers (●). During this phase, the orientation is also controlled relative to the green box. After grasping the box (●), the end-effector is moved to a position above of the yellow box (●), with the orientation also aligned to the yellow box. Subsequently, it is lowered a few centimeters (●) and the box is released. To avoid overturning the box stack, the end-effector is then moved up again (●) before moving on to the next box. This scheme is continued until the task is complete. Three of the seven position MPs approach the different boxes that are supposed to be stacked on the yellow box. The remaining four MPs all represent positions above of the yellow box with different heights. The different heights result from the growing size of the box stack and are consistent with the data.

For the box stacking task, we also compared our method against three other approaches. As baseline, we clustered the end-points of the segments using a HMM with GMMs as state emissions. Thus, in contrast to our approach, this approach does not take into account the entire segment and its direction. The second approach is the TP-GMM, introduced by Calinon et al. [47] and for instance used by Rozo et al. [26]. The TP-GMM is a hierarchical Gaussian Mixture Model with two layers. On the higher layer, each mixture represent an MP. On the lower layer, the mixtures represent the trajectory in the different task-spaces. The main difference to our approach is the semantic of an MP. While in our approach an MP describes a goal-directed movement, the TP-GMM models a joint distribution of the position and velocities. By conditioning on the current position, the model can also be used to generate desired velocities, allowing the model to be used for the reproduction

directly. The third approach is the BP-AR-HMM used by Niekum et al. [25] for decomposing a task into a set of MPs. Their approach uses the Beta-Process Autoregressive HMM for segmenting the demonstrations in the world frame. After training, they cluster the end-points of all segments assigned to a single MP in each coordinate frame. The cluster with the minimum variance is chosen as coordinate frame for the MP. In contrast to our approach, the baseline and the TP-GMM, the authors first segment the demonstrations in the world frame and later assign coordinate frames to the resulting MPs.

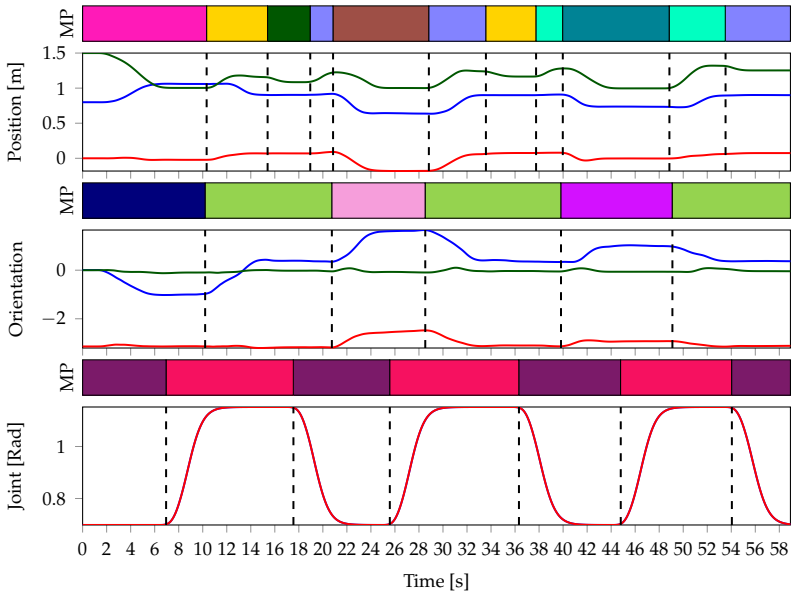



**Figure 9.** Data of one demonstration (in the world frame) and resulting task-decomposition for the box stacking task.

The decomposition resulted in a total of 13 MPs for our approach and the baseline approach, 9 MPs for the BP-AR-HMM and 65 MPs for the TP-GMM (for position, orientation and hand). The number of MPs for our approach, the baseline method and the TP-GMM has been determined using the Bayesian Information Criterion. For the BP-AR-HMM, we changed the hyperparameters of the method until we were satisfied with the results. A comparison of the resulting decompositions for all approaches is shown in Figure 10. While our approach clearly discriminates the steps the teacher performed for stacking the boxes (e.g., align green box over yellow box before stacking), the baseline approach merges some of these steps to a single MP. Even though the resulting MPs are similar to the MPs of our approach, they cannot be used for reproducing the task properly. The green box will not be stacked on the yellow box, but will rather be released above the yellow box. In addition, it may not be aligned properly. The green box may land correctly on the yellow box, but success will be rather random. The results show that it is not sufficient to cluster the end-points of the segments, as it becomes harder to assign the proper coordinate frames to the different segments. By incorporating the convergence properties of the segments, our approach yields better results compared to the baseline approach. The TP-GMM results in significantly more MPs compared to our approach for two reasons. First, as it models the whole trajectory instead of only the attractor goals, more MPs are needed in general. Second, if for instance an object is approached from two different directions, the two trajectories will be very different (even in the coordinate frame of the object), but their target position will be the same. Therefore, our approach is able to model such movements with fewer MPs compared to the TP-GMM. While the TP-GMM has the advantage of being able to model arbitrarily non-linear trajectories, the coordinate frame assignments made by our model better resemble the natural structure of the demonstrated task (see Figure 10b). We consider it future work to combine the advantages of both models. The BP-AR-HMM results in the fewest number of MPs. While the resulting coordinate frame

assignments in Figure 10b look promising at first, the phase where the red box is stacked on the yellow box is represented in the wrong coordinate frame. The reason is that the demonstrations are segmented in the world frame before assigning the coordinate frames to the individual MPs. As a result, segments which could be assigned to the same MP may be assigned to different MPs, as they are dissimilar in the world frame. Therefore, we believe that the coordinate frame assignment should be integrated into the segmentation process. We would also like to point out that none of the approaches chose the world frame for a task phase, which is a desired property for the box stacking task as all movements the teacher demonstrated were performed relative to one of the objects. Therefore, all approaches ignore the irrelevant world frame for the box stacking task.

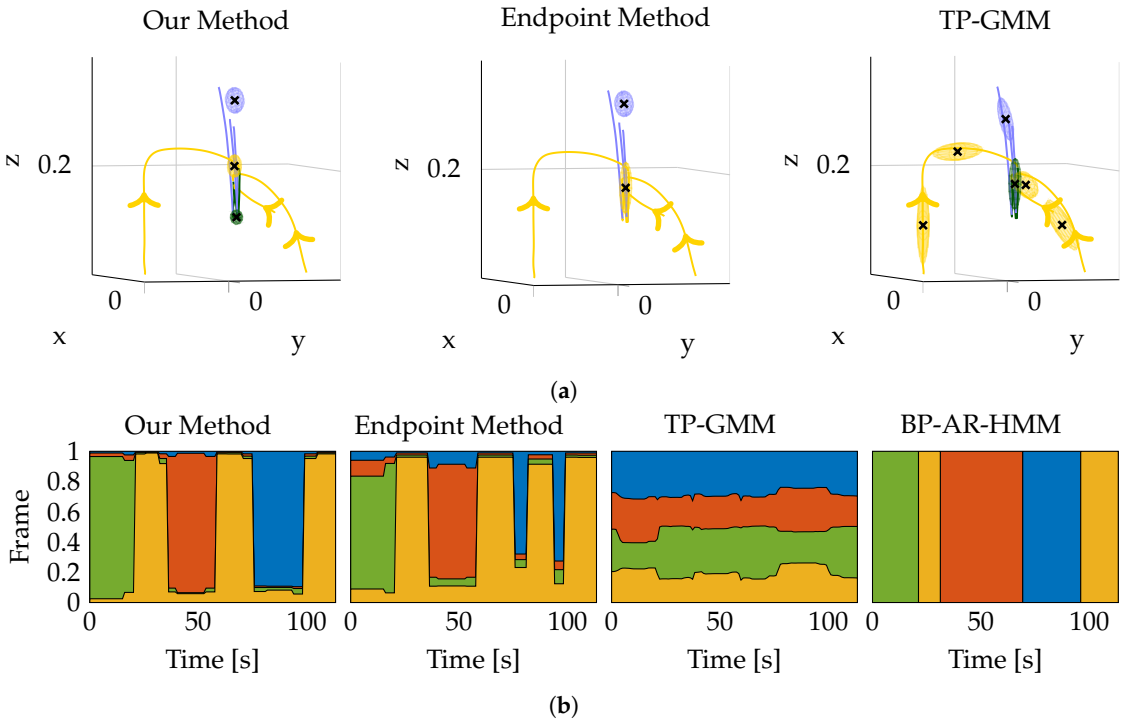



**Figure 10.** Comparison of our method with a baseline approach and the TP-GMM. For the baseline approach, we clustered the end-points of each segment using a HMM with GMMs as state emissions. Further discussions of the results can be found in Section 5.2. (**a**) In this snippet of the box stacking decomposition results, the green box was stacked on the yellow box. The decompositions are shown spatially in the coordinate frame of the yellow box. The colors correspond to the colors from Figure 9. Compared to our approach, the baseline approach merges the red and green MP to a single MP. The TP-GMM in general needs more MPs, as it models a path instead of MP targets. As it is difficult to visualize the parameters of the auto-regressive model, the BP-AR-HMM is not shown here; (**b**) For one full demonstration, the most likely coordinate frames over time are shown in the lower plots. The colors indicate the active coordinate frame (frame of the yellow, green, red or blue box). Our approach clearly distinguishes between phases of getting the different boxes and stacking them on the yellow box. The other approaches either do not separate the phases in such a clear way or (TP-GMM) or pick the wrong coordinate frame in some task phases (Endpoint method and BP-AR-HMM).

As a proof of concept, we reproduced the task on the real robot on a setup that was not demonstrated to the robot (see Figure 8). The reproduction showed that the robot can reproduce the task correctly. All MPs are represented in the correct coordinate frame and sequenced in the correct order. The boxes were successfully stacked on top of each other, even though they were not stacked as accurately as in the demonstrations. Most likely, the reason for the small displacement (approx. 0.5 cm to 1.0 cm) was due to the imprecision of the Kinect sensor used for tracking the objects.

### *5.3. Light Bulb Unscrewing*

For the unscrewing task, the human teacher demonstrated the following sequence of subtasks. First, he approached the light bulb with the end-effector. Subsequently, the robot's hand was closed. Next, the teacher rotated the wrist of the robot arm and unscrewed the bulb. During this movement, he also pulled the light bulb (by applying a force along the $z$-axis), to test whether it is still in its holder. After turning the light bulb, the fingers were opened and the wrist was rotated back. This unscrewing cycle was repeated until the light bulb got loose. Next, the light bulb was pulled out of the holder and the end-effector was moved to a box where the hand was opened again. For the demonstrations, we put the light bulb holder and the box to three different positions and performed three kinesthetic demonstrations for each setup. The setups are depicted in Figure 11. Two objects were tracked in the scene, the light bulb holder and the box.

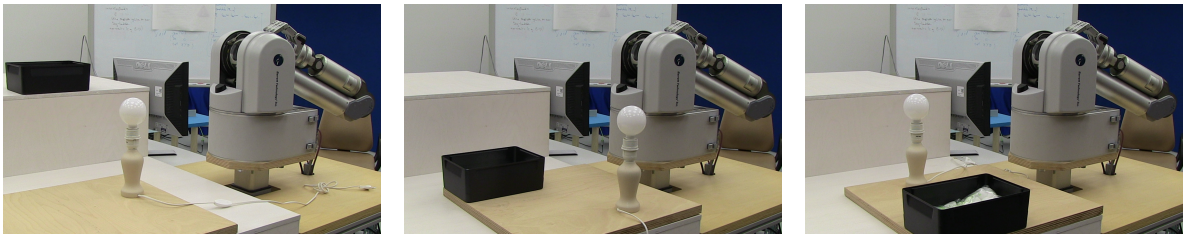


**Figure 11.** The three different experimental setups for the light bulb unscrewing task. For each setup, the light bulb holder and box were put to different locations on the two tables.

Our method extracted nine MPs from the demonstrations, as shown in Table 3. Four MPs control the position of the end-effector and one MP controls the position of the end-effector along the $x$- and $y$-axes, while applying a force along the $z$-axis. Additionally, two MPs control the orientation of the end-effector and two MPs the finger configuration. Figure 12 depicts the most likely MP for each point in time for one of the nine demonstrations. Except for small time gaps between changes of position, orientation and/or finger MPs, the MP sequences exactly reflect the subtask sequence the teacher performed and thus we can associate each MP with a meaningful description in Table 3. Please note that the time gaps are not incorrect but instead reflect the behavior of the teacher. For instance, it is not possible to unscrew the bulb before closing the fingers. Therefore, the teacher usually waits for a short moment until he or she is sure the fingers are closed before starting to unscrew, causing the aforementioned time gaps in the data.

Even though the number of unscrewing repetitions varied over the demonstrations, all demonstrations were decomposed in a similar way. We illustrated this in Figure 13, which shows all demonstrations spatially. As all subtasks performed by the teacher can be described relative to either the light bulb or the box, the world frame was not chosen for any MP controlling the position of the end-effector. During the demonstrations, the teacher was standing at a fixed position and aligned the orientations equally for all demonstrations. However, the objects and thus also their corresponding coordinate frames were rotated for the different setups. As a result, our algorithm chose the world frame for all orientation and finger configuration MPs.

For all position MPs, the resulting target forces are depicted in Table 4. Only for the ●-MP the target force along the $z$-axis is large enough so that the MP is force-controlled along this axis. The MP corresponds to the teacher pulling the light bulb while unscrewing. When reproducing the task, this force leads to an acceleration of the robot's arm as soon as the light bulb gets loose, enabling the system to detect the loose bulb without any external sensors. The drawback is that we cannot detect whether the light bulb is still in the hand of the robot or if it slipped during reproduction.

For the reproduction of the task, we set the light bulb holder to five different positions and performed two trials for each position. In all ten trials, the robot was able to unscrew the light bulb and successfully put it in the container box.

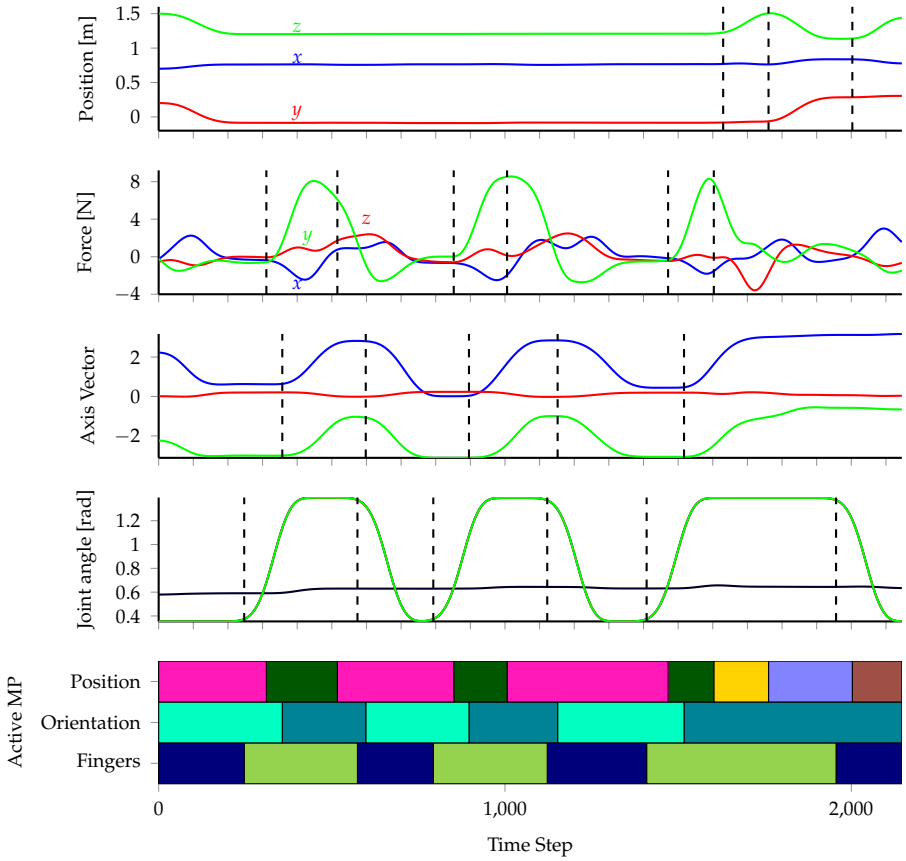
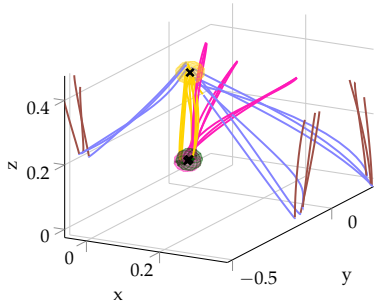


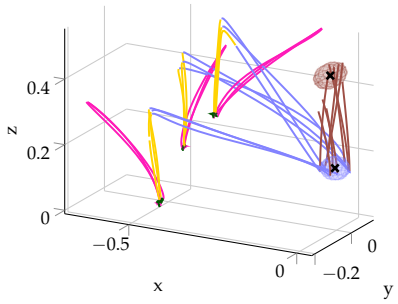

**Figure 12.** Task-decomposition over time for one of the nine demonstrations. From the top, the plots show the position of the end-effector, the measured forces at the wrist, the orientation of the end-effector and the finger joint angles, respectively. The dashed lines indicate the zero-velocity crossings and contact changes. The plot on the bottom shows the most likely MP for each point in time. All MPs can be associated with a meaningful description (see Table 3).

**Table 3.** Resulting coordinate frame and description for each MP. Only the MP that is active during unscrewing applies a force, as the teacher was pulling the light bulb during this phase of the task. Two MPs control the orientation of the end-effector and the fingers, respectively.

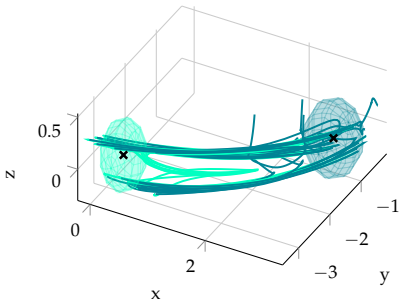
| MP | Description | Frame |
|---|---|---|
| ● | Approach Light Bulb | Light Bulb |
| ● | Pull Bulb (Force in z) | Light Bulb |
| ● | Lift Bulb | Light Bulb |
| ● | Approach Box | Box |
| ● | Approach Final Position | Box |
| ● | Screw | World |
| ● | Unscrew | World |
| ● | Open Fingers | World |
| ● | Close Fingers | World |



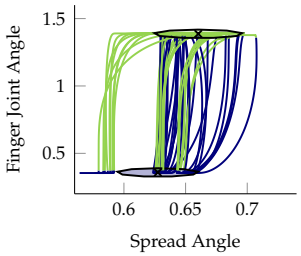

**(a)** Positions Light Bulb Frame




**(b)** Positions Box Frame




**(c)** Orientations World Frame




**(d)** Fingers World Frame

**Figure 13.** Task-space trajectories for all nine demonstrations of the light bulb unscrewing task. Plots (**a**,**b**) show the end-effector positions in the light bulb coordinate frame and box coordinate frame, respectively. The colors indicate to which MP each segment is assigned to. The markers correspond to the mean $\boldsymbol{\mu}$ of the MP targets and the ellipsoids indicate their covariance matrices $\boldsymbol{\Sigma}$. Please note that the position target of the blue MP is hard to see because it coincides with the target of the red MP. The targets are only plotted in the coordinate frame that was assigned to each MP. Plots (**c**,**d**) show the orientations of the end-effector and the finger configurations, respectively (both in the world frame). As for this task all three fingers were aligned equally, only the joint angle of one finger is shown in (**d**).

**Table 4.** Resulting target forces in Newton in the corresponding frames of the position MPs. Only the ●-MP is force-controlled along the $z$-axis because the desired force is large enough.

| MP | $x$ | $y$ | $z$ |
|---|---|---|---|
| ● | 0.08 | −1.43 | -0.51 |
| ● | −1.08 | 0.30 | **7.12** |
| ● | 1.08 | −1.35 | 0.02 |
| ● | 0.06 | 0.01 | −0.17 |
| ● | −0.24 | −0.85 | −0.88 |

### *5.4. Discussion of the Experiments*

The experimental results show that our method segments the demonstrations properly and finds a meaningful decomposition for all tasks. The reproduction showed that our system can learn to perform the tasks successfully. Still, the method has some properties that need to be discussed and some limitations that need further research.

As already mentioned, we scaled the force data before training to compensate for the noisy sensor and the different units (Newton vs. meters). Additionally, the scale factor reflects an important issue occurring in kinesthetic demonstrations. While it is easy to guide a gravity compensated robot to

a desired position, applying desired forces is difficult. For instance, in the light bulb task we tried to pull the bulb always with the same force. Still, the forces were roughly in a range of 5 to 15 N. Without scaling the force data, the method may incorrectly assign two pulling segments to different MPs, As they converge to very different force targets. We evaluated different scale factors and observed that the decomposition is the same for a broad range of values (1/20 to 1/80 yield the same results). Hence, there does not seem to be a need for fine-tuning the scale parameter for different tasks. Still, we state that from the force data of a kinesthetic demonstration, it is usually only clearly distinguishable whether the teacher wanted to push or pull in a certain direction or did not apply a force at all.

One limitation of the system is that the teacher has to know at least a little bit about the assumptions we make about the demonstrations. The teacher should be aware that he or she should pause between successive motions instead of co-articulating between them. The reason is that the decomposition relies on the segmentation, which uses ZVCs. Hence, no segments will be found if the end-effector does not stop between successive motions. Second, our MPs represent point-to-point motions in task-space. There is no need to move the end-effector in completely straight lines, but the teacher should at least roughly perform point-to-point motions. While this requirement seems to be very restrictive, a broad range of tasks can be performed using our framework. Still, we consider arbitrary trajectory shapes and co-articulation between successive motions interesting future research topics. An alternative to require the teacher to stop between successive motions would be to let the teacher tell the system about the transition points, such as proposed by Akgun et al. [48].

## 6. Conclusions

We presented an approach for learning sequential force skills from kinesthetic demonstrations. To learn a skill, the approach decomposes the task into a set of MPs and learns to sequence them. The core of the decomposition method is the Directional Normal Distribution (DND). The probability distribution allows simultaneously determining the most likely sequence of MPs as well as their composition, i.e., their coordinate frames, control variables and target coordinates from the demonstrations. Determining the control variables allows distinguishing phases of force from position control, enabling a robot to explicitly apply forces only when needed. The resulting MP sequences resemble very closely the natural structure of the task. The evaluation showed that our method successfully learns to perform three different tasks from fewer than ten demonstrations. In future work, we plan to extend our MP framework and the decomposition method, such that we can learn more sophisticated tasks that for instance require to follow a desired force profile or arbitrarily shaped movements.

**Author Contributions:** Conceptualization, S.M., M.G., J.K. and J.P.; Methodology, S.M., M.G., J.K. and J.P.; Software, S.M., M.G. and J.K.; Visualization, S.M.; Writing—original draft, S.M.; Writing—review & editing, S.M., M.G., J.K. and J.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AR | Augmented Reality |
| BIC | Bayesian Information Criterion |
| AR | Augmented Reality |
| BIC | Bayesian Information Criterion |
| DMP | Dynamic Movement Primitive |
| DND | Directional Normal Distribution |
| DoF | Degree of freedom |
| DS | Dynamical System |
| EM | Expectation-Maximization |

| | |
|---|---|
| GMM | Gaussian Mixture Model |
| HMM | Hidden Markov Model |
| IRL | Inverse Reinforcement Learning |
| KMP | Kernelized Movement Primitive |
| MP | Movement Primitive |
| TP-GMM | Task-Parametrized Gaussian Mixture Model |
| ZVC | Zero-velocity crossing |

## Appendix A. Constants for EM-Algorithm

For deriving the EM-algorithm, the integral

$$\int_0^\infty p\left(t\middle|\left[\boldsymbol{x},\boldsymbol{v}\right],\boldsymbol{\theta}\right)\log p\left(\left[\boldsymbol{x},\boldsymbol{v}\right],t\middle|\boldsymbol{\theta},\mu_t,\sigma_t\right)dt$$

has to be evaluated, which leads to

$$\begin{aligned}&=\int_0^\infty c_1 e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}\left(c_2(\boldsymbol{\theta})t^2+c_3(\boldsymbol{\theta})t+c_4(\boldsymbol{\theta})\right)dt\\&=c_1\left(c_2(\boldsymbol{\theta})d_1+c_3(\boldsymbol{\theta})d_2+c_4(\boldsymbol{\theta})d_3\right),\end{aligned}$$

where we used the constants

$$\begin{aligned}c_1&=\frac{1}{\sqrt{(2\pi)}\sigma_t}\frac{1}{1-\Phi(-\frac{\mu_t}{\sigma_t})},\\c_2&=-\frac{1}{2}\boldsymbol{v}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{v}-\frac{1}{2\sigma_t^2},\\c_3&=\frac{1}{2}\boldsymbol{v}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{x})+\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{v}+\frac{\mu_t}{\sigma_t^2},\\c_4&=-\frac{1}{2}\log\left((2\pi)^{d+1}\sigma_t^2\left|\boldsymbol{\Sigma}\right|\right)-\log\left(1-\Phi\left(-\frac{\mu_t}{\sigma_t}\right)\right)-\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{x})-\frac{\mu_t^2}{2\sigma_t^2}\\d_1&=\mu_t\sigma_t^2 e^{-\frac{\mu_t^2}{2\sigma_t^2}}+\sqrt{\frac{\pi}{2}}\sigma_t\left(\mu_t^2+\sigma_t^2\right)\left(1+\operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right),\\d_2&=\sigma_t^2 e^{-\frac{\mu_t^2}{2\sigma_t^2}}+\sqrt{\frac{\pi}{2}}\mu_t\sigma_t\left(1+\operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right),\\d_3&=\sqrt{\frac{\pi}{2}}\sigma_t\left(1+\operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right).\end{aligned}$$

For more information, the reader is referred to Section 3.1.

## Appendix B. Constants for Orientations

The start orientation rotated with an angle $t$ around the angular velocity axis $\boldsymbol{a}=[a_x,a_y,a_z]^T$, given in the inertia frame can be defined as

$$\boldsymbol{R}_{\mathrm{VI}}(t)=\boldsymbol{R}_{\mathrm{VS}}(t)\boldsymbol{R}_{\mathrm{SI}},$$

where the entries of the matrix $\boldsymbol{R}_{\mathrm{VS}}(t)$ are

$$\begin{aligned}
\boldsymbol{R}_{\mathrm{VS}}^{(0,0)}(t) &= (1-a_x^2)\cos(t) + a_x^2,\\
\boldsymbol{R}_{\mathrm{VS}}^{(0,1)}(t) &= -a_x a_y \cos(t) + a_z \sin(t) + a_x a_y,\\
\boldsymbol{R}_{\mathrm{VS}}^{(0,2)}(t) &= -a_x a_z \cos(t) - a_y \sin(t) + a_x a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(1,0)}(t) &= -a_x a_y \cos(t) - a_z \sin(t) + a_x a_y,\\
\boldsymbol{R}_{\mathrm{VS}}^{(1,1)}(t) &= (1-a_y^2)\cos(t) + a_y^2,\\
\boldsymbol{R}_{\mathrm{VS}}^{(1,2)}(t) &= -a_y a_z \cos(t) + a_x \sin(t) + a_y a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(2,0)}(t) &= -a_x a_z \cos(t) + a_y \sin(t) + a_x a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(2,1)}(t) &= -a_y a_z \cos(t) - a_x \sin(t) + a_y a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(2,2)}(t) &= (1-a_z^2)\cos(t) + a_z^2,
\end{aligned}$$

with $\boldsymbol{R}_{\mathrm{VS}}^{(i,j)}(t)$ being the entry of the $i$th row and $j$th column. The matrix can be written in the form $\boldsymbol{R}_{\mathrm{VS}}(t) = \boldsymbol{D}\cos(t) + \boldsymbol{E}\sin(t) + \boldsymbol{F}$ with

$$\begin{aligned}
\boldsymbol{D} &= \begin{pmatrix} 1-a_x^2 & -a_x a_y & -a_x a_z \\ -a_x a_y & 1-a_y^2 & -a_y a_z \\ -a_x a_z & -a_y a_z & 1-a_z^2 \end{pmatrix},\\
\boldsymbol{E} &= \begin{pmatrix} 0 & a_z & -a_y \\ -a_z & 0 & a_x \\ a_y & -a_x & 0 \end{pmatrix},\\
\boldsymbol{F} &= \begin{pmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{pmatrix}.
\end{aligned}$$

Therefore, we can write

$$\begin{aligned}
\boldsymbol{R}_{\mathrm{VI}}(t) &= \boldsymbol{R}_{\mathrm{VS}}(t)\boldsymbol{R}_{\mathrm{SI}}\\
&= (\boldsymbol{D}\cos(t) + \boldsymbol{E}\sin(t) + \boldsymbol{F})\\
&= \boldsymbol{D}\boldsymbol{R}_{\mathrm{SI}}\cos(t) + \boldsymbol{E}\boldsymbol{R}_{\mathrm{SI}}\sin(t) + \boldsymbol{F}\boldsymbol{R}_{\mathrm{SI}}.
\end{aligned}$$

The values of the constant vectors $\boldsymbol{d}$, $\boldsymbol{e}$ and $\boldsymbol{f}$ used in (11) can then be found by multiplying the matrices and vectorizing the resulting matrices.

*Appendix B.1. Axis Angle Derivation*

The axis angle between the rotated coordinate system and the target coordinate system is defined as

$$\theta(\phi_\omega) = \cos^{-1}\left(\frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{R}_{FV}) - 1\right)\right) \tag{A1}$$

$$= \cos^{-1}\left(\frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{R}_{FS}(\boldsymbol{R}_{VS})^T) - 1\right)\right) \tag{A2}$$

We are only interested in the trace of the resulting transformation matrix $\boldsymbol{R}_{FV} = \boldsymbol{R}_{FV}^{(0,0)} + \boldsymbol{R}_{FV}^{(1,1)} + \boldsymbol{R}_{FV}^{(2,2)}$, which can be written as follows

$$
\begin{aligned}
\boldsymbol{R}_{FV}^{(0,0)} &= \boldsymbol{R}_{FS}(0,0)(a_x^2(1-\cos(t))+\cos(t)) \\
&\quad + \boldsymbol{R}_{FS}(0,1)(a_x a_y(1-\cos(t)) - a_z\sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(0,2)(a_x a_z(1-\cos(t)) + a_y\sin(t)), \\
\boldsymbol{R}_{FV}^{(1,1)} &= \boldsymbol{R}_{FS}(1,0)(a_x a_y(1-\cos(t)) + a_z\sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(1,1)(a_y^2(1-\cos(t))+\cos(t)) \\
&\quad + \boldsymbol{R}_{FS}(1,2)(a_y a_z(1-\cos(t)) - a_x\sin(t)), \\
\boldsymbol{R}_{FV}^{(2,2)} &= \boldsymbol{R}_{FS}(2,0)(a_x a_z(1-\cos(t)) - a_y\sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(2,1)(a_y a_z(1-\cos(t)) + a_x\sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(2,2)(a_z^2(1-\cos(t))+\cos(t)).
\end{aligned}
$$

The terms can be sorted and rearranged

$$
\begin{aligned}
\boldsymbol{R}_{FV}^{(0,0)} &= a_0\cos(t) + b_0\sin(t) + c_0, \\
\boldsymbol{R}_{FV}^{(1,1)} &= a_1\cos(t) + b_1\sin(t) + c_1, \\
\boldsymbol{R}_{FV}^{(2,2)} &= a_2\cos(t) + b_2\sin(t) + c_2.
\end{aligned}
$$

Here, we used

$$
\begin{aligned}
a_0 &= \boldsymbol{R}_{FS}^{(0,0)}(1-a_x^2) - \boldsymbol{R}_{FS}^{(0,1)}a_x a_y - \boldsymbol{R}_{FS}^{(0,2)}a_x a_z, \\
a_1 &= \boldsymbol{R}_{FS}^{(1,1)}(1-a_y^2) - \boldsymbol{R}_{FS}^{(1,0)}a_x a_y - \boldsymbol{R}_{FS}^{(1,2)}a_y a_z, \\
a_2 &= \boldsymbol{R}_{FS}^{(2,2)}(1-a_z^2) - \boldsymbol{R}_{FS}^{(2,0)}a_x a_z - \boldsymbol{R}_{FS}^{(2,1)}a_y a_z, \\
b_0 &= \boldsymbol{R}_{FS}^{(0,1)}a_z - \boldsymbol{R}_{FS}^{(0,2)}a_y, \\
b_1 &= \boldsymbol{R}_{FS}^{(1,2)}a_x - \boldsymbol{R}_{FS}^{(1,0)}a_z, \\
b_2 &= \boldsymbol{R}_{FS}^{(2,0)}a_y - \boldsymbol{R}_{FS}^{(2,1)}a_x. \\
c_0 &= \boldsymbol{R}_{FS}^{(0,0)}a_x^2 + \boldsymbol{R}_{FS}^{(0,1)}a_x a_y + \boldsymbol{R}_{FS}^{(0,2)}a_x a_z \\
c_1 &= \boldsymbol{R}_{FS}^{(1,0)}a_x a_y + \boldsymbol{R}_{FS}^{(1,1)}a_y^2 + \boldsymbol{R}_{FS}^{(1,2)}a_y a_z \\
c_2 &= \boldsymbol{R}_{FS}^{(2,0)}a_x a_z + \boldsymbol{R}_{FS}^{(2,1)}a_y a_z + \boldsymbol{R}_{FS}^{(2,2)}a_z^2
\end{aligned}
$$

Using $a = a_0 + a_1 + a_2$, $b = b_0 + b_1 + b_2$ and $c = c_0 + c_1 + c_2$, the axis angle becomes

$$
\theta(t) = \cos^{-1}\left(\frac{1}{2}\left(a\ \cos(t) + b\sin(t) + c - 1\right)\right).
$$

This axis angle representation is used in Section 3.2 to compute a minimum angle $t$ for the current estimate of the parameters.

## References


1. Manschitz, S.; Kober, J.; Gienger, M.; Peters, J. Probabilistic Decomposition of Sequential Force Interaction Tasks into Movement Primitives. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016.
2. Manschitz, S.; Kober, J.; Gienger, M.; Peters, J. Learning to Sequence Movement Primitives from Demonstrations. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014.

3. Manschitz, S.; Kober, J.; Gienger, M.; Peters, J. Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robot. Auton. Syst.* **2015**, *74*, 97–107, doi:/10.1016/j.robot.2015.07.005. [CrossRef]
4. Lioutikov, R.; Neumann, G.; Maeda, G.; Peters, J. Probabilistic Segmentation Applied to an Assembly Task. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Korea, 3–5 November 2015.
5. Lemme, A.; Reinhart, R.F.; Steil, J.J. Self-supervised bootstrapping of a movement primitive library from complex trajectories. In Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots, Madrid, Spain, 18–20 November 2014; doi:10.1109/HUMANOIDS.2014.7041443. [CrossRef]
6. Endres, D.; Christensen, A.; Omlor, L.; Giese, M.A. Emulating Human Observers with Bayesian Binning: Segmentation of Action Streams. *ACM Trans. Appl. Percept.* **2011**, *8*, 16:1–16:12. [CrossRef]
7. Krishnan, S.; Garg, A.; Patil, S.; Lea, C.; Hager, G.; Abbeel, P.; Goldberg, K. Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning. *Int. J. Robot. Res.* **2015**. [CrossRef]
8. Lafferty, J.D.; McCallum, A.; Pereira, F.C.N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001), Williamstown, MA, USA, 28 June–1 July 2001.
9. Baisero, A.; Mollard, Y.; Lopes, M.; Toussaint, M.; Lutkebohle, I. Temporal Segmentation of Pair-Wise Interaction Phases in Sequential Manipulation Demonstrations. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015.
10. Lee, S.H.; Suh, I.H.; Calinon, S.; Johansson, R. Autonomous framework for segmenting robot trajectories of manipulation task. *Auton. Robots* **2015**, *38*, 107–141, doi:10.1007/s10514-014-9397-9. [CrossRef]
11. Huang, B.; Li, M.; De Souza, R.L.; Bryson, J.J.; Billard, A. A modular approach to learning manipulation strategies from human demonstration. *Auton. Robots* **2016**, *40*, 903–927, doi:10.1007/s10514-015-9501-9. [CrossRef]
12. Figueroa, N.; Billard, A. A Physically-Consistent Bayesian Non-Parametric Mixture Model for Dynamical System Learning. In Proceedings of the 2nd Annual Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; Volume 87, pp. 927–946.
13. Gräve, K.; Behnke, S. Incremental action recognition and generalizing motion generation based on goal-directed features. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 Ocober 2012.
14. Kulić, D.; Ott, C.; Lee, D.; Ishikawa, J.; Nakamura, Y. Incremental learning of full body motion primitives and their sequencing through human motion observation. *Int. J. Robot. Res.* **2012**, *31*, 330–345. [CrossRef]
15. Shiarlis, K.; Wulfmeier, M.; Salter, S.; Whiteson, S.; Posner, I. TACO: Learning Task Decomposition via Temporal Alignment for Control. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
16. Konidaris, G.; Kuindersma, S.; Grupen, R.; Barto, A. Robot learning from demonstration by constructing skill trees. *Int. J. Robot. Res.* **2012**, *31*, 360–375. [CrossRef]
17. Niekum, S.; Osentoski, S.; Atkeson, C.G.; Barto, A.G. Online Bayesian Changepoint Detection for Articulated Motion Models. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015.
18. Guerra-Filho, G.; Aloimonos, Y. A Language for Human Action. *Computer* **2007**, *40*, 42–51, doi:10.1109/MC.2007.154. [CrossRef]
19. Meier, F.; Theodorou, E.; Schaal, S. Movement Segmentation and Recognition for Imitation Learning. *J. Mach. Learn. Res.* **2012**, *22*, 761–769.
20. Gutzeit, L.; Kirchner, E.A. Automatic Detection and Recognition of Human Movement Patterns in Manipulation Tasks. In Proceedings of the 3rd International Conference on Physiological Computing Systems, Lissabon, Portugal, 27–28 July 2016; pp. 54–63.
21. Gutzeit, L.; Fabisch, A.; Petzoldt, C.; Wiese, H.; Kirchner, F. Automated Robot Skill Learning from Demonstration for Various Robot Systems. In *KI 2019: Advances in Artificial Intelligence*; Springer: Cham, Switzerland, 2019; pp. 168–181, doi:10.1007/978-3-030-30179-8_14. [CrossRef]

22. Ranchod, P.; Rosman, B.; Konidaris, G. Nonparametric Bayesian Reward Segmentation for Skill Discovery Using Inverse Reinforcement Learning. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015.
23. Michini, B. Bayesian Nonparametric Reward Learning from Demonstration. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2013.
24. Krishnan, S.; Garg, A.; Liaw, R.; Thananjeyan, B.; Miller, L.; Pokorny, F.T.; Goldberg, K. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *Int. J. Robot. Res.* **2019**, *38*, 126–145. [CrossRef]
25. Niekum, S.; Osentoski, S.; Konidaris, G.D.; Chitta, S.; Marthi, B.; Barto, A.G. Learning Grounded Finite-State Representations from Unstructured Demonstrations. *Int. J. Robot. Res.* **2015**, *34*, 131–157. [CrossRef]
26. Rozo, L.; Calinon, S.; Caldwell, D.G.; Jiménez, P.; Torras, C. Learning Physical Collaborative Robot Behaviors From Human Demonstrations. *IEEE Trans. Robot.* **2016**, *32*, 513–527, doi:10.1109/TRO.2016.2540623. [CrossRef]
27. Maeda, G.J.; Neumann, G.; Ewerton, M.; Lioutikov, R.; Kroemer, O.; Peters, J. Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks. *Auton. Robots* **2017**, *41*, 593–612, doi:10.1007/s10514-016-9556-2. [CrossRef]
28. Caccavale, R.; Saveriano, M.; Finzi, A.; Lee, D. Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction. *Auton. Robots* **2019**, *43*, 1291–1307, doi:10.1007/s10514-018-9706-9. [CrossRef]
29. Wu, H.; Xu, Z.; Yan, W.; Su, Q.; Li, S.; Cheng, T.; Zhou, X. Incremental Learning Introspective Movement Primitives From Multimodal Unstructured Demonstrations. *IEEE Access* **2019**, *7*, 159022–159036. [CrossRef]
30. Huang, Y.; Rozo, L.; Silvério, J.; Caldwell, D.G. Kernelized movement primitives. *Int. J. Robot. Res.* **2019**, *38*, 833–852, doi:10.1177/0278364919846363. [CrossRef]
31. Gao, X.; Ling, J.; Xiao, X.; Li, M. Learning Force-Relevant Skills from Human Demonstration. *Complexity* **2019**, *2019*, 1–11, doi:10.1155/2019/5262859. [CrossRef]
32. Abu-Dakka, F.J.; Nemec, B.; Jørgensen, J.A.; Savarimuthu, T.R.; Krüger, N.; Ude, A. Adaptation of manipulation skills in physical contact with the environment to reference force profiles. *Auton. Robots* **2015**, *39*, 199–217, doi:10.1007/s10514-015-9435-2. [CrossRef]
33. Ijspeert, A.J.; Nakanishi, J.; Schaal, S. Movement imitation with nonlinear dynamical systems in humanoid robots. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; pp. 1398–1403, doi:10.1109/ROBOT.2002.1014739. [CrossRef]
34. Steinmetz, F.; Montebelli, A.; Kyrki, V. Simultaneous kinesthetic teaching of positional and force requirements for sequential in-contact tasks. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Korea, 3–5 November 2015, doi:10.1109/HUMANOIDS.2015.7363552. [CrossRef]
35. Ureche, A.; Umezawa, K.; Nakamura, Y.; Billard, A. Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations. *IEEE Trans. Robot.* **2015**, *31*, 1458–1471, doi:10.1109/TRO.2015.2495003. [CrossRef]
36. Kober, J.; Gienger, M.; Steil, J. Learning Movement Primitives for Force Interaction Tasks. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015.
37. Pastor, P.; Kalakrishnan, M.; Righetti, L.; Schaal, S. Towards Associative Skill Memories. In Proceedings of the 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, 29 November–1 December 2012.
38. Mühlig, M.; Gienger, M.; Steil, J.J.; Goerick, C. Automatic selection of task spaces for imitation learning. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; doi:10.1109/IROS.2009.5353894. [CrossRef]
39. Kappler, D.; Pastor, P.; Kalakrishnan, M.; Wuthrich, M.; Schaal, S. Data-Driven Online Decision Making for Autonomous Manipulation. In Proceedings of the Robotics: Science and Systems, Rome, Italy, 13–17 July 2015.
40. Ekvall, S.; Kragic, D. Learning Task Models from Multiple Human Demonstrations. In Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN), Hatfield, UK, 6–8 September 2006; pp. 358–363.

41. Nicolescu, M.N.; Mataric, M.J. Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice. In Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia, 14–18 July July 2003.
42. Pardowitz, M.; Zollner, R.; Dillmann, R. Learning sequential Constraints of Tasks from User Demonstrations. In Proceedings of the 5th IEEE-RAS International Conference on Humanoid Robots, Tsukuba, Japan, 5 December 2005.
43. Medina, J.R.; Billard, A. Learning Stable Task Sequences from Demonstration with Linear Parameter Varying Systems and Hidden Markov Models. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017.
44. Flanagan, J.R.; Bowman, M.C.; Johansson, R.S. Control strategies in object manipulation tasks. *Curr. Opin. Neurobiol.* **2006**, *16*, 650–659. [CrossRef]
45. Higham, N. Matrix Nearness Problems and Applications. In *Applications of Matrix Theory*; Oxford University Press: Oxford, UK, 1989; pp. 1–27.
46. Garrido-Jurado, S.; noz Salinas, R.M.; Madrid-Cuevas, F.; Marín-Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.* **2014**, *47*, 2280–2292, doi:10.1016/j.patcog.2014.01.005. [CrossRef]
47. Calinon, S.; Li, Z.; Alizadeh, T.; Tsagarakis, N.G.; Caldwell, D.G. Statistical dynamical systems for skills acquisition in humanoids. In Proceedings of the 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, 29 November–1 December 2012.
48. Akgun, B.; Cakmak, M.; Jiang, K.; Thomaz, A.L. Keyframe-based Learning from Demonstration. *Int. J. Soc. Robot.* **2012**, *4*, 343–355, doi:10.1007/s12369-012-0160-0. [CrossRef]